TUGAS AKHIR – KS141501

RANCANG BANGUN APLIKASI UNTUK MENGEKSTRAKSI DAN MENYAJIKAN INFORMASI EVENT SECARA REALTIME DARI MEDIA SOSIAL DENGAN METODE CONDITIONAL RANDOM FIELD MENGGUNAKAN KERANGKA KERJA FLASK PYTHON

APPLICATION FOR EXTRACTING AND VISUALIZING EVENT INFORMATION IN REALTIME FROM SOCIAL MEDIA USING CONDITIONAL RANDOM FIELD USING FLASK PYTHON FRAMEWORK

DEWANGGA PRASETYA PRAJA
NRP 0521 14 400 0038

Dosen Pembimbing
Renny Pradina Kusumawardani, S.T., M.T., SCJP

Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman ini sengaja dikosongkan

TUGAS AKHIR – KS141501

RANCANG BANGUN APLIKASI UNTUK MENGEKSTRAKSI DAN MENYAJIKAN INFORMASI EVENT SECARA REALTIME DARI MEDIA SOSIAL DENGAN METODE CONDITIONAL RANDOM FIELD MENGGUNAKAN KERANGKA KERJA FLASK PYTHON

DEWANGGA PRASETYA PRAJA
NRP 0521 14 4000 0038

Dosen Pembimbing:
Renny Pradina Kusumawardani, S.T., M.T., SCJP

Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman ini sengaja dikosongkan

FINAL PROJECT – KS141501

APPLICATION FOR EXTRACTING AND VISUALIZING EVENT INFORMATION IN REALTIME FROM SOCIAL MEDIA USING CONDITIONAL RANDOM FIELD WITH FLASK PYTHON FRAMEWORK

DEWANGGA PRASETYA PRAJA
NRP 0521 14 4000 0038

Supervisor:
Renny Pradina Kusumawardani, S.T., M.T., SCJP

Departement of Information System
Faculty of Information Communication and Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI UNTUK MENGEKSTRAKSI DAN MENYAJIKAN INFORMASI EVENT SECARA *REALTIME* DARI MEDIA SOSIAL DENGAN METODE *CONDITIONAL RANDOM FIELD* MENGGUNAKAN KERANGKA KERJA FLASK PYTHON

TUGAS AKHIR
Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

DEWANGGA PRASETYA PRAJA
NRP. 0521 14 4000 0038

Surabaya, 16 Juli 2018

KEPALA
DEPARTEMEN SISTEM INFORMASI

Ir. Aris Tjahyanto, M.Kom
NIP 19650310 199102 1 001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI UNTUK MENGEKSTRAKSI DAN MENYAJIKAN INFORMASI EVENT SECARA *REALTIME* DARI MEDIA SOSIAL DENGAN METODE *CONDITIONAL RANDOM FIELD* MENGGUNAKAN KERANGKA KERJA FLASK PYTHON

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:

DEWANGGA PRASETYA PRAJA
NRP. 0521 14 4000 0038

Disetujui Tim Penguji: Tanggal Ujian: 10 Juli 2018
Periode Wisuda: September 2018

Renny Pradina Kusumawardani, S.T., M.T., SCJP. — **(Pembimbing I)**

Faizal Johan Atletiko, S.Kom., M.T — **(Penguji I)**

Radityo Prasetianto Wibowo, S.Kom, M.Kom — **(Penguji II)**

Halaman ini sengaja dikosongkan

RANCANG BANGUN APLIKASI UNTUK MENGEKSTRAKSI DAN MENYAJIKAN INFORMASI EVENT SECARA *REALTIME* DARI MEDIA SOSIAL DENGAN METODE *CONDITIONAL RANDOM FIELD* MENGGUNAKAN KERANGKA KERJA FLASK PYTHON

Nama Mahasiswa	**: Dewangga Prasetya Praja**
NRP	**: 0521 14 4000 0038**
Jurusan	**: Sistem Informasi FTIK-ITS**
Pembimbing 1	**: Renny Pradina Kusumawardani S.T., M.T., SCJP**

ABSTRAK

Melihat tingginya jumlah pengguna internet di dunia, mendorong masyarakat dunia untuk terlibat dalam berbagai jenis media sosial. Beberapa sosial media yang populer digunakan sekarang ini antara lain adalah Instagram, Facebook dan Twitter. Media sosial memberikan pengaruh besar terhadap penyebaran informasi. Hal ini mendorong penyelenggara event untuk bekerja sama dengan media partner dalam mempromosikan kegiatan melalui sosial karena lebih efisien dan ekonomis dibanding melalui media konvensional. Di sisi lain, publikasi merupakan salah satu faktor penting dalam kesuksesan suatu acara khususnya untuk acara yang mengandalkan jumlah pengunjung sebagai sumber utama pendapatan. Semakin banyak tiket yang terjual semakin besar pula pemasukan. Informasi kegiatan dapat lebih optimal jika informasi pokok, seperti waktu, tempat, nama kegiatan, maupun narahubung, dapat diorganisir dengan lebih baik. Salah satu metode dalam mengorganisir informasi dari data tidak terstruktur menjadi data terstruktur adalah menggunakan information extraction atau lebih spesifik pada event extraction. Dengan menggunakan event extraction, informasi seperti nama tempat dapat dikenali dan ditandai. Terutama

untuk informasi tempat dapat ditandai pada peta, atau waktu pelaksanaan kegiatan dapat diubah menjadi tanggal dengan format yang standar.

Maka dari itu, diperlukan adanya aplikasi yang dapat mengorganisir informasi event yang diberikan oleh media partner. Secara teknis, algoritma yang digunakan dalam penelitian ini adalah menggunakan Conditional Random Field yang didukung menggunakan PoS Tagging dan Named Entity Recognition. Dalam proses pelaksanaan, terdapat lima komponen utama yang dibuat, antara lain komponen Scrapping, Sistem Pelabelan, Model CRF, Sistem Kafka, dan Visualisasi. Serta dilakukan unit testing pada modul yang dianggap krusial.

Hasil dari penelitian ini didapatkan kesimpulan bahwa nilai akurasi secara keseluruhan model adalah 94,48%. Namun nilai tersebut didominasi oleh label OTHER yang memiliki jumlah yang jauh dari label yang lain. Walaupun memiliki nilai akurasi yang tinggi, disisi lain nilai recall yang didapat untuk label NAME dan INFO hanya mencapai 49% dan 60%.

kata kunci: Information Extraction, Media Sosial, Conditional Random Field, Place Recognition, Time Recognition

APPLICATION FOR EXTRACTING AND VISUALIZING EVENT INFORMATION IN REALTIME FROM SOCIAL MEDIA USING CONDITIONAL RANDOM FIELD WITH FLASK PYTHON FRAMEWORK

Nama Mahasiswa	**: Dewangga Prasetya Praja**
NRP	**: 0521 14 4000 0038**
Jurusan	**: Sistem Informasi FTIK-ITS**
Pembimbing 1	**: Renny Pradina Kusumawardani S.T., M.T., SCJP**

ABSTRACT

High amount of internet user in the world encourage peoples to involve in every kind of social media. Social medias that recently popular were Instagram, Facebook and Twitter. Social medias give big impacts toward the dissemination of information. These impacts encourage event organizers to collaborate with event media partners in term of promoting their event via social media because this way makes more efficient and lower budget than via conventional media. On the other hand, publication is one of important factor on the success rate of an event, especially for some sort of event that depend on the amount of visitors as source of income. More ticket sold means the more income that would get. Event information would became more optimal if main information – time, place, name of event, nor contact person – could be organized well. Method to organize those informations is by using information extraction for event – Event Extraction. Informations can be recognized and labeled. Especially, the place where the event were held can be located on the map, or the time when the event were held can be formatted as standard format.

Therefore, application that could organize event information from social media is needed. Technically, this research is using Conditional Random Field as the main method supported by PoS Tagging and Named Entity Recognition. There are five main components in the implementation – Scrapping, Tagging

System, CRF Model, Kafka System, and Visualization. There are unit testing on some important modul in every components. The result of this research can be concluded that the accuracy value is 94,48%. However, this precentage were dominated by OTHER label that has higher amount of data than the other labels. Eventhough the accuracy is high, on the other hand, recall value from the NAME label and INFO label only reach 49% and 60%.
keywords: Information Extraction, Social Media, Conditional Random Field, Place Recognition, Time Recognition

KATA PENGANTAR

Puji dan syukur penulis tuturkan kehadirat Allah SWT, Tuhan Semesta Alam yang telah memberikan kekuatan dan hidayah-Nya kepada penulis sehingga penulis mendapatkan kelancaran dalam menyelesaikan tugas akhir ini yang merupakan salah satu syarat kelulusan pada Departemen Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember.

Terima kasih penulis sampaikan kepada pihak-pihak yang telah mendukung, memberikan saran, motivasi, semangat, tercapainya tujuan pembuatan tugas akhir ini. Tugas akhir ini tidak akan pernah terwujud tanpa bantuan dan dukungan dari berbagai pihak yang telah meluangkan waktu, tenaga, pikiran, dan doanya. Secara khusus penulis akan menyampaikan ucapan terima kasih banyak kepada:

1. Bapak Budi Mulyanto dan Ibu Puji Handayani selaku kedua orang tua, serta Sofia Rasyida Nugraheni selaku saudara kandung dari penulis yang tiada henti memberikan dukungan semangat serta doa.
2. Ibu Renny Pradina Kusumawardani, S.T., M.T., SCJP, selaku dosen pembimbing yang senantiasa meluangkan waktu, membagikan ilmu, memberikan petunjuk, dan memotivasi penulis sepanjang pelaksanaan tugas akhir ini.
3. Bapak Radityo Prasetianto W., S.Kom., M.Kom. dan Bapak Faizal Johan Atletiko, S.Kom, MT., selaku dosen penguji yang telah memberikan kritik dan saran dalam perbaikan tugas akhir.
4. Seluruh dosen Jurusan Sistem Informasi ITS yang telah memberikan ilmu yang bermanfaat kepada penulis.

5. Alden Delfian Wattimena, Prasetyo Wahtyu, Anugrah Dwi Putra, Adrian Afnandika, Muhammad Arif Samudro, BQ Zuyyina Hilya, dan Hans Juno Panjaitan, selaku teman seperjuangan serta satu bimbingan yang telah memberikan dukungan, berbagi ilmu, serta menemani hingga akhir pengerjaan tugas akhir ini.
6. Sofia Rasyida Nugraheni, Fandhi Akhmad, Prasetyo Wahyu, Redian Galih Irianti, Ammar Fauzan, dan Alden Delfian Wattimena selaku orang yang telah bersedia membantu melakukan pelabelan data yang digunakan pada penelitian ini.
7. Fandhi Akhmad, Tubagus Hendro Pramono, Naufal Raihan Noly, Brilianto Wilis Satria Nugraha, Indra Alfarhizi, Muhammad Iqbal Imaduddin, Umar Abdul Aziz, Aloysius Tatus Kristanto, Nur Rochman Darmawan, Muhammad Arif Samudro, Rizky Bintang Orlando, Sigit Tri Baskoro, dan Faiz Nur Fitrah Insani selaku teman satu E-HOME yang selalu menemani penulis dalam waktu suka maupun duka.
8. Tubagus Hendro Pramono selaku asisten laboratorium ADDI yang telah menemani dan membantu memfasilitasi saat menghuni laboratorium ADDI.
9. Rekan-rekan HMSI Muda Berkarya, HMSI Kolaborasi, Kementerian SOSMAS BEM ITS, Sobat Bumi Surabaya, dan OSIRIS yang telah memberikan pengalaman serta kenangan semasa kuliah.
10. Aprilia Rizky Subagya yang telah memberikan dorongan moral dan semangat.
11. Berbagai pihak yang tidak bisa disebutkan satu persatu yang telah turut serta membantu dalam kelancaran pelaksanaan tugas akhir ini dari awal hingga akhir.

Penyusunan laporan ini masih jauh dari kata sempurna sehingga penulis mengharapkan adanya kritik saran serta pengembangan lebih lanjut di masa yang akan datang. Semoga buku tugas akhir ini dapat memberikan manfaat bagi pembaca.

Surabaya, 10 Juli 2018

Penulis,

Dewangga Prasetya Praja

Halaman ini sengaja dikosongkan

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR KODE

Halaman ini sengaja dikosongkan

DAFTAR PERSAMAAN

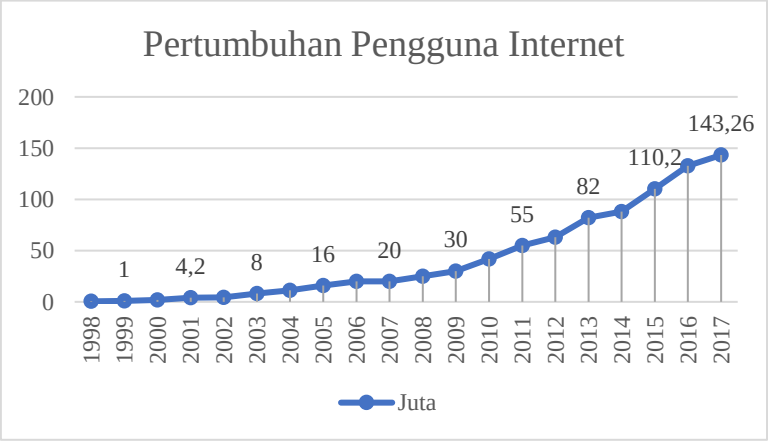
Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1. Latar Belakang

Publikasi merupakan salah satu faktor penting dalam kesuksesan suatu acara. Terlebih lagi untuk suatu acara yang mengandalkan jumlah pengunjung sebagai sumber utama pendapatan. Semakin banyak tiket yang terjual semakin besar pula pemasukan.

Berdasarkan survei dari Asosiasi Penyelenggara Jasa Internet Indonesia (APJII), peningkatan jumlah pengguna internet dari tahun ke tahun meningkat dengan cepat hingga pada tahun 2017 terdapat 143 juta dari 262 juta orang penduduk Indonesia [1] seperti ditunjukkan pada Gambar 1.1. Melihat pertumbuhan pengguna internet dari tahun ke tahun maka internet memberikan berbagai manfaat dan perubahan gaya hidup pada masyarakat Indonesia. Layanan sosial media menjadi salah satu layanan yang sering diakses oleh pengguna internet Indonesia, yaitu hingga mencapai 87,13% [1].



Gambar 1.1 Pertumbuhan pengguna internet dari tahun 1998 hingga 2017

Di masa serba teknologi sekarang ini, persebaran informasi sudah sangat cepat, terutama di sektor dunia maya. Di berbagai sosial media sudah banyak fitur yang mendukung untuk diadakan suatu kegiatan, entah dalam bentuk undangan untuk skala besar maupun kecil. Selain itu, bermunculan banyak media partner yang memberikan informasi tentang kegiatan atau *event*. Sosial media menjadi salah satu jalur yang dimanfaatkan dalam menyebarkan informasi.

Walaupun sosial media merupakan sarana yang mudah dimanfaatkan, akan tetapi sosial media memiliki kelemahan yaitu informasi yang diberikan tidak bisa dikelola dengan baik. Sering kita melihat bahwa kiriman yang sama dilakukan berulang-ulang dalam jangka waktu tertentu. Hal ini dinilai kurang efisien. Selain itu, pencarian informasi berdasarkan tempat dan tanggal pelaksanaan pun tidak dimungkinkan. Pengguna perlu melakukan pemindaian satu per satu dari kiriman terbaru hingga yang terdahulu.

Pada penelitian kali ini, peneliti akan membangun suatu aplikasi yang mampu menandai faktor-faktor penting dalam kiriman media partner. Teknologi yang digunakan pada penelitian ini memanfaatkan *Natural Language Processing* yang lebih spesifik pada *Information Extraction*. Penelitian Teknologi berdasarkan *Natural Language Processing* sudah cukup populer dan memberikan banyak perubahan dalam gaya hidup manusia. Metode yang populer digunakan untuk melakukan *Information Extraction* adalah *Named Entity Recognition* [2][3][4].

Dalam beberapa penelitian sebelumnya menunjukkan banyaknya metode dalam melakukan *Named Entity Recognition.* Namun salah satu metode yang populer adalah *Conditional Random Field.*

1.2. Perumusan Masalah

1. Bagaimana cara melakukan akuisisi data event melalui media sosial?

2. Bagaimana metode yang tepat dalam mengolah teks tidak terstruktur yang ada media sosial menjadi informasi yang terstruktur?
3. Bagaimana cara melakukan visualisasi data yang telah diolah agar dan memiliki *user experience* yang baik?

1.3. Batasan Masalah

1. Media sosial yang akan digunakan dalam penelitian adalah Facebook dan Instagram.
2. Data yang diambil hanya pada akun yang sering membagikan informasi tentang acara atau kegiatan yang akan diadakan di kota Surabaya.
3. Aplikasi yang dibuat dapat mengenali minimal waktu dan lokasi diadakan kegiatan tersebut.

1.4. Tujuan Penelitian

Tujuan akhir dari penelitian ini adalah mendapatkan aplikasi web berbasis *natural language processing* yang mampu memberikan informasi terkini secara terstruktur tentang *event* yang diambil dari akun media sosial dari beberapa media partner.

1.5. Manfaat Penelitian

1.5.1. Bagi Peneliti

Manfaat bagi peneliti antara lain adalah mendapatkan pengalaman dan pengetahuan pengembangan aplikasi yang menggunakan *Natural Language Processing*, serta dapat dijadikan sebagai karya yang dapat dibanggakan.

1.5.2. Bagi Penyelenggara Acara

Manfaat bagi penyelenggara acara mendapatkan manfaat berupa peningkatan jumlah partisipan pada acara yang diselenggarakan.

1.6. Relevansi

Topik yang saya angkat memiliki korelasi dengan mata kuliah Sistem Cerdas dan Algoritma Struktur Data, serta Penggalian Data dan Analitika Bisnis. Topik yang saya ambil merupakan

salah satu fokus utama di Laboratorium Akuisisi Data dan Diseminasi Informasi, yaitu lebih spesifik pada *Natural Language Processing* yang ada pada bagian Diseminasi Informasi.

BAB II
TINJAUAN PUSTAKA

2.1. Penelitian Sebelumnya

Pada *paper* yang dibuat oleh Shasha Liao dan Ralph Grishman [5] yang berjudul *"Using Document Level Cross Event Inference to Improve Event Extraction."* Dalam penelitian tersebut mencari tipe kejadian pada suatu kalimat berdasarkan *entity*, *entity mention*, *timex*, *event mention*, *event trigger*, dan *event mention arguments*. Dari penelitian tersebut dapat disimpulkan bahwa penggunaan model *Document level* dan *Cross Event* dapat meningkatkan performa sistem ekstraksi kejadian untuk menganalisis pada level kalimat.

Pada paper berjudul *"Event Extraction on Indonesian News Article Using Multiclass Categorization"* oleh Masayu Leylia Khodra [6] melakukan penelitian tentang pencarian informasi terstruktur pada teks tidak terstruktur atau teks tidak terstruktur, atau lebih dikenal dengan ekstraksi informasi. Ekstraksi yang dilakukan spesifik pada ekstraksi kejadian artikel berita berbahasa Indonesia. Pada penelitian tersebut digunakan metode Klasifikasi *Multiclass* untuk menentukan 5W1H dari 90 artikel berita dari berbagai laman berita terkenal di Indonesia. Dari 90 berita dilakukan *labeling* pada setiap komponen 5W1H yang dilakukan oleh 3 *anotator*. Setelah melakukan *labeling* pada artikel kemudian dilakukan pencarian pola dengan melakukan pada setiap kata pada 5W1H yang didapatkan sebelumnya. Label yang dilakukan menggunakan awalan, selipan serta akhiran untuk masing-masing 5W1H, yang selanjutnya akan disebut sebagai *token*. Untuk algoritma yang ingin dibandingkan adalah *Adaboost* dan C4.5 untuk dijadikan metode pembelajaran. Dari performa kedua algoritma tersebut secara keseluruhan C4.5 lebih unggul daripada *Adaboost*.

Sama halnya dengan *paper* dengan judul *"Event Extraction from Indonesian Tweets using Conditional Random Field"* oleh Fawwas Muhammad dan Masayu Leylia Khodran [3]

melakukan penelitian lanjutan namun dengan objek sosial media Twitter yang berdomain bahasa Indonesia. Metode yang digunakan pun berbeda yaitu menggunakan *Conditional Random Field* yang merupakan algoritma *linear regression* algoritma yang cocok untuk melakukan *Named Entity Recognition*. Label yang digunakan tidak terdiri dari 5W1H namun lebih spesifik pada nama, tempat, waktu, serta info kegiatan yang dipublikasikan di Twitter. Eksperimen tersebut bertujuan untuk mendapatkan konfigurasi terbaik untuk arsitektur sistem cerdas. Hasil dari eksperimen ditemukan bahwa diperlukan modul filter dan pengekstraksi sebagai komponen utama sistem karena diketahui bahwa perlu adanya pemisah antara *tweet* yang relevan dan yang tidak relevan.

2.2. Dasar Teori

2.2.1. Natural Language Processing

Natural Language Processing (NLP) adalah suatu bidang ilmu yang memanfaatkan penggunaan perangkat lunak dalam melakukan pengolahan bahasa alami, seperti ucapan dan teks. NLP memanfaatkan penggunaan bidang ilmu linguistik seperti tata bahasa, semantik, dan fonetik. Dengan adanya suatu pola dan aturan dalam bahasa maka penggunaan metode matematis dalam memahami bahasa alami dapat dilakukan [7]. Namun pengolahan bahasa bukan hal yang mudah dikarenakan kombinasi dari berbagai kosakata maupun aturan yang kompleks dalam suatu bahasa. Maka dari itu, penggunaan perangkat lunak dapat dilakukan untuk melakukan banyak perhitungan sekaligus secara cepat.

Dengan memahami pola serta aturan penggunaan bahasa, maka bukan suatu hal yang tidak mungkin untuk diterapkan pada mesin agar mesin tersebut dapat berpikir secara linguistik layaknya manusia. Sehingga mesin dapat membantu manusia dalam berbagai bidang dengan cara berkomunikasi dengan pengguna.

Berikut ini adalah berbagai penerapan NLP [8]:

- *Spell and Grammar Checking*
- *Word Prediction*
- *Information Retrieval*
- *Text Categorization*
- *Summarization*
- *Question Answering*
- *Information Extraction*
- *Machine Translation*
- *Sentiment Analysis*
- *Optical Character Recognition*
- *Speech recognition*
- *Speech synthesis*
- *Spoken dialog systems*

a. Information Extraction

Information Extraction (IE) adalah salah satu penerapan NLP yang berguna untuk melakukan ekstraksi pada informasi penting dari suatu teks untuk dikumpulkan dan diletakkan pada suatu pola terstruktur [8].

Salah satu contoh penerapan dari Information Extraction adalah pada halaman web Wikipedia.

b. Event Extraction

Event Extraction adalah suatu langkah yang digunakan untuk mencari *event* di mana suatu entitas berpartisipasi [9].

Dalam mengetahui kapan suatu event dalam teks itu terjadi perlu melakukan *temporal expression recognition*, yaitu entitas yang berhubungan dengan waktu seperti nama hari, bulan hari libur, dan lain sebagainya, bisa juga entitas waktu yang bersifat relatif seperti “selama dua hari”, “hari senin hingga kamus”, “malam ini”, “lusa”, “minggu depan” dan lain-lain [9].

Langkah selanjutnya adalah *temporal expression normalization* yaitu melakukan pemetaan dari entitas waktu relatif tersebut ke dalam tanggal dan waktu yang spesifik [9].

Langkah terakhir adalah *template filling* yaitu mencari suatu situasi dalam teks kemudian mengisi suatu *template* sesuai yang dibutuhkan, seperti contoh pada Tabel 2.1 di bawah ini:

Tabel 2.1 Contoh template filling dari event extraction

Pesan : Sunday Application Day 2018 Pameran Pendidikan Australia, UK, & Ireland 18 Maret 2018 TS SUITES SURABAYA Join the Overseas Study Expo with AUG Student Services Open for Public! Info: 087853236436
Nama Kegiatan : Sunday Application Day 2018
Waktu : 18 Maret 2018
Tempat : TS SUITES SURABAYA
Narahubung : 087853236436

c. Named Entity Recognition

Named Entity Recognition (NER) adalah salah satu langkah penting yang digunakan untuk mencari dan mengklasifikasikan entitas dalam suatu teks [4][9]. NER merupakan langkah pertama dalam melakukan *Information Extraction.*

Algoritma standar dari NER berupa pemberian label secara berurutan dari kata per kata yang diberikan penanda (*tag*) berupa batas dan tipenya. Tipe dapat berupa nama orang, organisasi, lokasi, waktu, entitas politik, ataupun merek produk [9]. Contoh pemberian label dapat dilihat pada Tabel 2.2 di bawah ini.

Tabel 2.2 Contoh IOB tagging

TOKEN	IOB LABEL
Presiden	B-PERSON
Joko	I-PERSON
Widodo	I-PERSON
akan	O
melakukan	O
kunjungan	O
kerja	O
ke	O
Negara	B-LOCATION
Afganistan	I-LOCATION
Senin	B-TIME
lalu	I-TIME

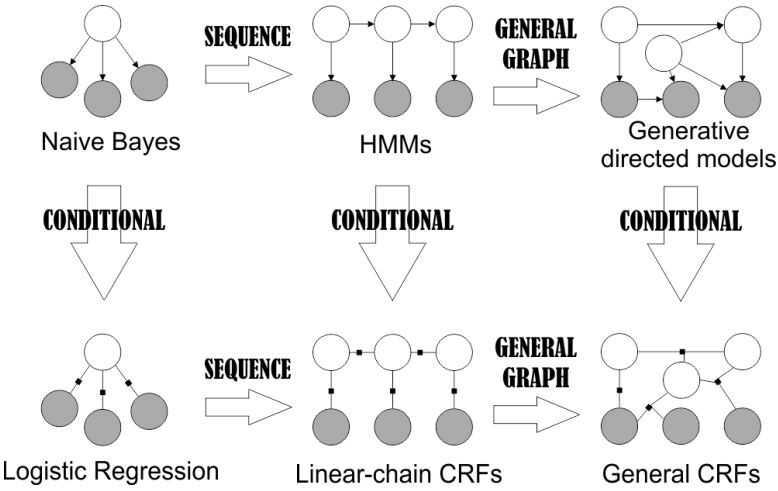
IOB Tagging adalah digunakan untuk menandakan B sebagai awal *(beginning)* dan I sebagai isi *(inside)* dari setiap tipe entitas. Sedangkan O digunakan sebagai penanda bahwa kata tersebut berada di luar *(outside)* dari semua entitas yang disediakan [9].

2.2.2. Conditional Random Field

Conditional Random Field (CRF) adalah suatu kerangka kerja untuk membangun model probabilistik untuk melakukan segmentasi data dan pelabelan data berurutan [10].

CRF muncul dengan adanya ide bahwa dalam *Named Entity Recognition,* diasumsikan suatu *part-of-speech* (POS) *tag* atau *named entity tag* bergantung pada kondisi kata itu sendiri, serta kata sebelum dan setelah kata tersebut [10].

Berikut adalah gambar model hubungan antara beberapa jenis algoritma yang memiliki kesamaan pada Gambar 2.1[11]:



Gambar 2.1 Hubungan antara Naive Bayes, Logistic Regression, HMM, linear-chain CRF, generative model, dan general CRF

Pada CRF, setiap fungsi fitur adalah fungsi dengan *input* sebagai berikut [12]:

- Sebuah kalimat ditandai sebagai s
- Posisi i dari suatu kata pada sebuah kalimat
- Label l_i dari kata tersebut
- Label l_{i-1} dari kata sebelumnya

Berikut adalah fungsi bagaimana menghitung bobot fitur dari semua kata pada kalimat yang ditunjukkan pada Persamaan 2.1 [12].

$$score(l|s) = \sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s, i, l_i, l_{i-1})$$

Persamaan 2.1 Bobot fitur

Kemudian disubstitusikan ke fungsi perubahan bobot fitur pada Persamaan 2.2 [12].

$$p(l|s) = \frac{\exp[score(l|s)]}{\sum_{l'} \exp[score(l'|s)]}$$

$$= \frac{\exp[\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s, i, l_i, l_{i-1})]}{\sum_{l'} \exp[\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s, i, l'_i, l'_{i-1})]}$$

Persamaan 2.2 Perubahan bobot fitur

2.2.3. Python

Python adalah salah satu bahasa pemrograman multifungsi yang bersifat *open-source* [13]. Python bisa digunakan secara fleksibel karena mendukung pemrograman berbasis objek, prosedural, atau fungsional. Versi Python yang stabil pada Februari 2018 adalah 3.6.4 [14].

a. CRFsuite

CRFsuite adalah implementasi dari metode Conditional Random Fields untuk melakukan pelabelan data secara sekuensial. Dari berbagai implementasi CRF lainnya, CRF memiliki beberapa keunggulan dan perbedaan sebagai berikut[15].

- Pelatihan model dan pelabelan yang cepat dibanding yang lain[16].
- Format data yang sederhana untuk pelatihan dan uji coba model.
- Menggunakan *linear-chain* (*first-order* Markov) CRF.
- Evaluasi performa yang dilakukan berbarengan dengan proses pelatihan model.
- File format yang efisien dalam melakukan penyimpanan dan akses model CRF.

CRFsuite awalnya dikembangkan menggunakan bahasa pemrograman C++. Namun dengan teknologi SWIG API, CRFsuite dapat dimanfaatkan dalam berbagai bahasa pemrograman lain, salah satunya adalah Python. *Library* CRFsuite yang ada di Python adalah *python-crfsuite*.

b. Natural Language Toolkit

Natural Language Toolkit (NLTK) adalah salah satu platform berbasis Python yang digunakan membuat program menggunakan bahasa manusia [17]. Versi NLTK yang stabil pada Februari 2018 adalah 3.2.5. NLTK menyediakan berbagai jenis *library* dalam melakukan berbagai tahapan dalam NLP seperti klasifikasi, *tokenization, stemming, tagging, parsing, semantic reasoning,* dam lain sebagainya. Selain itu, NLTK memiliki dokumentasi yang mudah dibaca dan contoh yang jelas menurut sepengetahuan dari penulis. Khusus untuk metode CRF, NLTK menggunakan modul turunan yaitu *sklearn-crfsuite* yang merupakan bentuk sederhana dari *python-crfsuite*.

c. L-BFGS Training Algorithm

Pelatihan yang dilakukan oleh NLTK menggunakan algoritma L-BFGS[18]. Berikut adalah training options yang dapat digunakan pada CRFTagger.

- 'feature.minfreq': Minimum frekuensi dari fitur.
- 'feature.possible_states': Memaksakan untuk menggenerate status fitur yang memungkinkan
- 'feature.possible_transitions': Memaksakan untuk menggenerate perubahan fitu yang memungkinkan
- 'c1': Koefisien untuk L1 *regularization*.
- 'c2': Koefisien untuk L2 *regularization*.
- 'max_iterations': Jumlah maksimal iterasi untuk optimasi L-BFGS
- 'num_memories': Jumlah batasan memori untuk memperkirakan matrix hessian inverse.
- 'epsilon': Epsilon untuk menguji konvergensi dari tujuan.
- 'period': Durasi iterasi untuk menghentikan percobaan.
- 'delta': Batasan untuk menghentikan percobaan; Pada L-BFGS iterasi berhenti jika peningkatan suatu log memungkinkan mencapai melebihi iterasi pada

periode yang terakhir, maka iterasi tersebut tidak akan melebihi batasan ini.

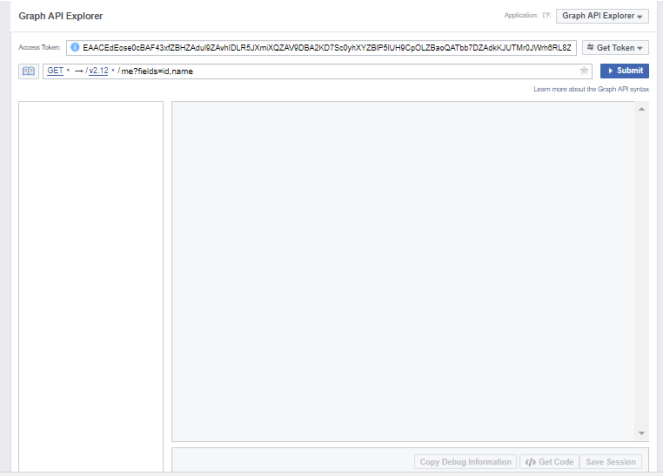
- 'linesearch': Jenis algoritma pencarian baris pada L-BFGS:
    - 'MoreThuente': More and Thuente's method,
    - 'Backtracking': Backtracking method dengan kondisi regular Wolfe,
    - 'StrongBacktracking': Backtracking method dengan kondisi strong Wolfe.
- 'max_linesearch': Jumlah maksimal percobaan untuk algoritma pencarian baris.

2.2.4. Facebook

Facebook adalah salah satu layanan jaringan sosial yang digunakan lebih dari 750 juta orang dari berbagai belahan dunia [19]. Facebook memberikan berbagai peluang bagi komunitas profit maupun non-profit dalam mengumpulkan banyak orang untuk saling berhubungan dan saling berbagi informasi dengan berbagai cara.

Facebook memberikan fitur bagi pengembang yang ingin menggunakan data milik Facebook dalam mengembangkan aplikasi. Fitur tersebut adalah *Graph API*. *Graph API* adalah salah satu cara untuk mengirimkan data ke dalam dan/atau keluar dari basis data Facebook [20]. *Graph API* menggunakan API berbasis HTTP tingkat rendah dalam melakukan *query* data dengan respons berupa format JSON.

Graph API juga menyediakan *tool* yang memudahkan pengembang untuk melakukan pengujian. Berikut adalah tampilan dari Graph API Explorer pada Gambar 2.2 di bawah ini:



Gambar 2.2 Tampilan User Interface dari Graph API Explorer

Berikut adalah contoh dalam melakukan permintaan melalui *graph API* pada Kode 2.1.

```
GET /v2.12/eventsurabaya/feed HTTP/1.1
Host: graph.facebook.com
/eventsurabaya/feed?fields=id,created_time,message,attachmen
ts
```

Kode 2.1 Cara melakukan permintaan melalui Facebook Graph API

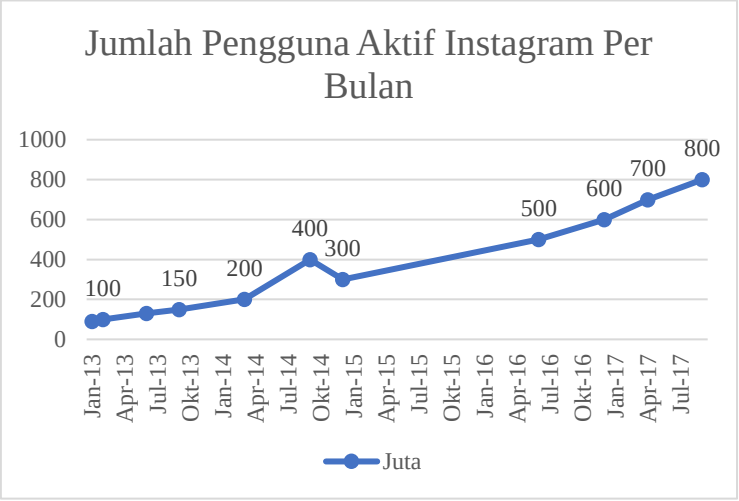
Peneliti akan memilih beberapa akun yang memiliki kriteria sebagai media partner yang memberikan info kegiatan atau *event* dalam lingkup Surabaya.

2.2.5. Instagram

Instagram adalah salah satu platform media sosial yang berfokus pada aktivitas berbagi foto dan video. Instagram didesain untuk pengguna *smartphone* yang beringinan mengambil foto dari perangkat dan mengunggah gambar tersebut langsung melalui aplikasi Instagram [21].

Pengguna Instagram mencapai 800 juta pengguna aktif pada bulan September 2017 lalu dan terus bertambah hingga saat ini [22]. Dilihat tingginya jumlah peningkatan jumlah pengguna

seperti pada Gambar 2.3, Instagram dianggap sebagai salah satu alat pemasaran yang baik, sehingga muncul banyak merek yang berlomba-lomba menggunakan Instagram menjadi salah satu strategi bisnisnya.



Gambar 2.3 Jumlah pengguna aktif instagram per bulan dari Juni 2013 hingga September 2018

Peneliti akan memilih beberapa akun yang memiliki kriteria sebagai media partner yang memberikan info kegiatan atau *event* dalam lingkup Surabaya.

2.2.6. Twitter

Twitter adalah suatu platform media sosial berupa mikroblog yang memberikan fasilitas untuk mengirimkan kiriman berupa tulisan dengan panjang maksimal 280 karakter. Twitter berdiri pada tahun 2006 dan terus berkembang hingga pada puncaknya mencapai 95 juta *tweet* per hari pada September 2010 [23].

Twitter merupakan salah satu media sosial yang unik karena keterbatasan jumlah karakter pada setiap *tweet*. Hal ini mempengaruhi cara melakukan kiriman pengguna yang berbeda dengan media sosial lain. Kiriman yang ada di Twitter

cenderung tidak basa-basi dan tepat sasaran. Teks yang singkat padat dan jelas lebih mudah dalam melakukan pelabelan secara manual karena kata yang dibaca lebih sedikit dan lebih mudah dipahami manusia.

2.2.7. MongoDB

MongoDB adalah salah satu penyedia basis data yang bersifat NoSQL atau non relasional. Basis data non relasional sendiri berarti basis data yang tidak terikat dengan aturan yang ketat seperti relasi antar tabel, sehingga struktur penyimpanan data dapat lebih fleksibel dan dinamis. Berbeda dengan basis data pada umumnya yang berbentuk tabel, MongoDB terdiri dari dokumen yang bernama *collection* dan *data records*. Versi MongoDB yang stabil pada Februari 2018 adalah 3.6 [24].

2.2.8. Apache Kafka

Apache Kafka adalah platform yang digunakan untuk melakukan pengambilan data *streaming* secara *real-time*. Kafka dijalankan sebagai sebuah *cluster* pada server dan mampu mencakup beberapa *data center* sekaligus. *Kafka cluster* menyimpan rekaman *data stream* dalam kategori yang disebut topik. Setiap rekaman terdiri dari *key*, *value* dan *timestamp*.

Kafka terdiri dari empat komponen penting [25]:

- Producer API komponen yang berguna untuk menerbitkan rekaman data *stream* untuk satu atau lebih topik Kafka.
- Consumer API komponen yang digunakan untuk berlangganan pada satu atau lebih topik Kafka.
- Stream API komponen yang digunakan sebagai prosesor *stream* yang mengubah *input stream* dari topik kafka menjadi *output stream*.
- Connector API komponen yang digunakan untuk menjalankan produser atau konsumer yang terhubung dengan topik pada aplikasi atau sumber data, seperti basis data relasional.

2.2.9. Flask

Flask adalah salah satu kerangka kerja mikro dengan bahasa pemrograman Python yang digunakan untuk membuat web. Versi Flask yang stabil pada Mei 2018 adalah 1.0.2 [26].

Berbeda dengan Django yang sama-sama berbasis Python, Flask memiliki struktur yang lebih sederhana dan tidak memiliki fitur selengkap Django. Namun dalam hal pembuatan tampilan keduanya tidak jauh berbeda karena sama-sama menggunakan Jinja2 [26].

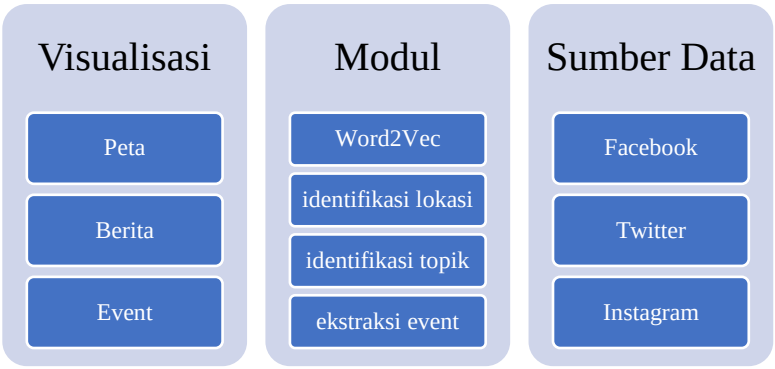
2.2.10. Google Calendar API

Google Calendar API digunakan untuk mengintegrasikan aplikasi yang dibuat dengan Google Calendar. Dengan memanfaatkan API ini aplikasi mampu menambahkan *event* pada kalender yang ada pada akun google pengguna secara otomatis.

2.2.11. CitiVis

CitiVIs merupakan aplikasi visualisasi keadaan terkini dari kota Surabaya yang diambil berdasarkan data media sosial. CitiVis terdiri dari berbagai modul NLP antara lain pengubahan bahasa informal menjadi formal (Word2Vec), identifikasi topik, identifikasi lokasi, ekstraksi *event.* Modul tersebut digabungkan dan mengambil data dari sosial media Facebook, Twitter dan Instagram secara *real-time* menggunakan Apache Kafka.

Pada penelitian ini, akan dibuat modul untuk melakukan ekstraksi *event* yang akan digunakan untuk pengembangan aplikasi CitiViz. Berikut adalah komponen dari aplikasi CitiVis pada Gambar 2.4 di bawah ini:

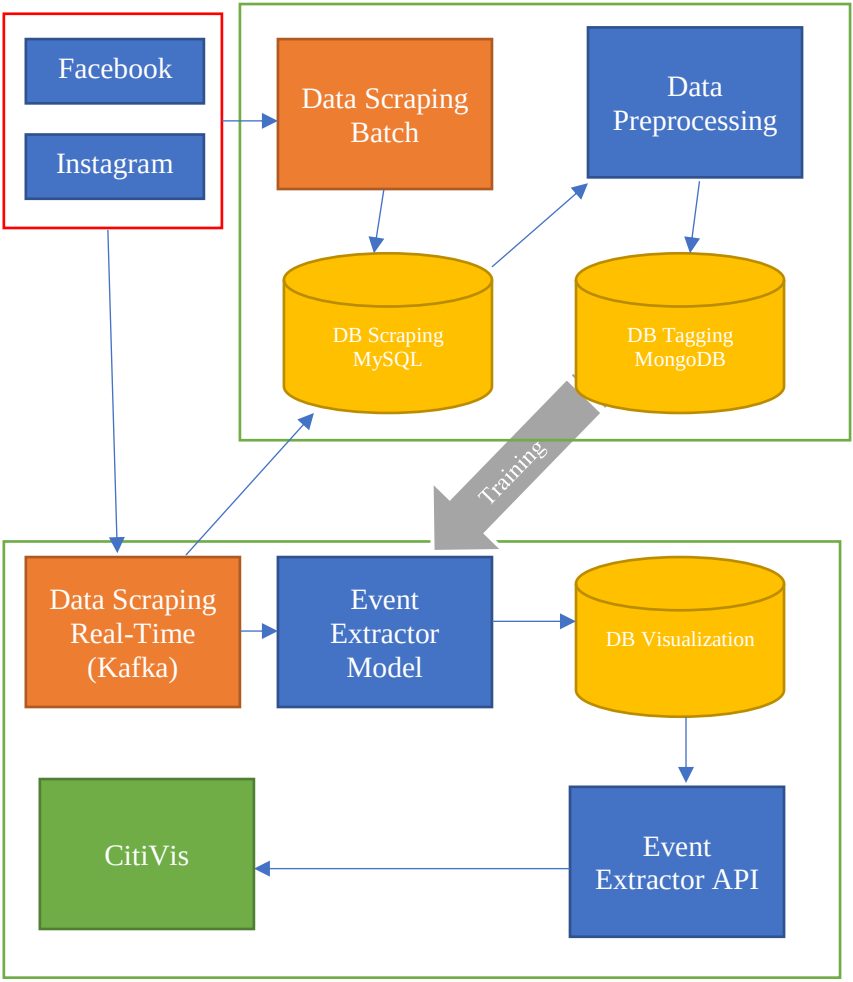


Gambar 2.4 Komponen CitiVis

BAB III
METODOLOGI

3.1. Rancangan Arsitektur Sistem

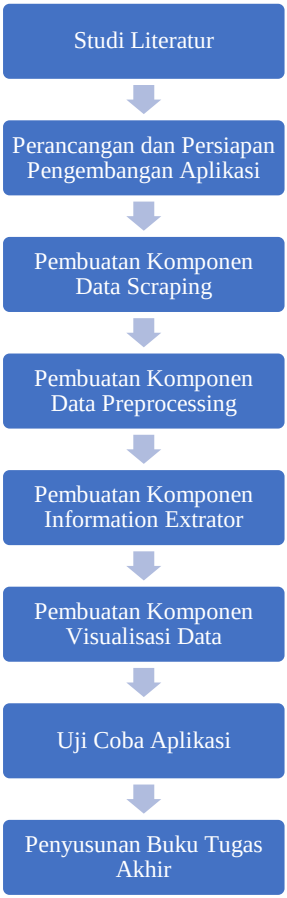
Berikut adalah diagram rancangan arsiteksur sistem pada Gambar 3.1.



Gambar 3.1 Rancangan arsitektur sistem

3.2. Tahap Pelaksanaan Tugas Akhir

Berikut adalah alur pelaksanaan tugas akhir pada Gambar 3.2.



Gambar 3.2 Metodologi pengerjaan tugas akhir

3.2.1. Studi Literatur

Studi literatur adalah kegiatan yang dilakukan dalam mempelajari teknis pembuatan serta kerangka kerja yang akan digunakan dalam tugas akhir ini. Studi literatur yang dilakukan bersumber dari dokumentasi, buku, *paper* penelitian terdahulu, beserta sumber *online* yang terkait.

3.2.2. Perencanaan dan Persiapan Pengembangan Aplikasi

Perencanaan pengembangan aplikasi bertujuan untuk mempersiapkan kebutuhan dalam pengembangan aplikasi antara lain berupa instalasi aplikasi serta *dependency* yang akan digunakan dalam *development* serta mencari *hosting* yang tepat untuk dilakukan implementasi.

3.2.3. Pembuatan Komponen Data Scraping

Data Scraping digunakan untuk mengambil data yang ada di sosial media untuk kemudian diolah dan dianalisis sebelumnya. *Scraping* dilakukan secara otomatis menggunakan Apache Kafka sebagai pengambilan data secara *real-time*. Sedangkan untuk data awal dilakukan secara agregat menggunakan Python.

Setelah pengambilan data dilakukan, maka tahap selanjutnya adalah pemilahan antara konten yang relevan dengan yang tidak relevan. Data relevan dalam hal ini adalah data yang memberikan informasi tentang *event.* Pada sistem ini perlu adanya klasifikasi antara konten tersebut. Maka dari itu perlu adanya proses pembuatan model yang dapat membedakan antara konten yang berhubungan dengan *event* maupun bukan.

Pembuatan model akan dilakukan mulai dari pembuatan *data training* dan *data testing*, serta pelatihan model dan uji coba model.

3.2.4. Pembuatan Komponen Data Preprocessing

Data Preprocessing digunakan dalam mengubah data mentah yang didapat dari proses *scrapping* menjadi data yang siap untuk dilakukan analisis. *Preprocessing* data terdiri dari

berbagai tahapan berikut antara lain: Normalisasi kata, *Tokenization*, *PoS Tagging*, *Named Entity Tagging*, Penyimpanan data preprocessing akan disimpan pada basis data mongoBD.

3.2.5. Pembuatan Komponen Information Extractor

Data yang sudah dilakukan *preprocessing* akan dianalisis lebih lanjut untuk diambil informasi mengenai acara yang akan berlangsung melalui proses *training data*. Proses *training data* akan dilakukan menggunakan NLTK menggunakan *library crfsuite*. Informasi yang akan disimpan terbatas pada 5W1H yang lebih spesifik dan fokus pada tempat, waktu, serta nama acara. Informasi tersebut akan dimasukkan dalam *database* beserta *url* gambar, serta sumber asli informasi diambil.

3.2.6. Pembuatan Komponen Visualisasi Data

Dari sistem yang telah dibuat sebelumnya, peneliti akan mengambil data yang telah diolah untuk ditampilkan dalam bentuk web dengan fitur yang dapat memfilter tempat dan waktu kegiatan. Pengambilan data dari basis data dan model dilakukan melalui API yang dibuat menggunakan kerangka kerja Flask.

3.2.7. Uji Coba Aplikasi

Uji coba aplikasi akan dilakukan pada kode program yang mencakup *unit testing*. Testing dilakukan menggunakan library unittesting yang ada di python. Sedangkan pada Flask difokuskan pada HTTP testing dan database testing saja. Selain itu, dilakukan juga uji coba rilis minimal selama tiga hari pada setiap komponen untuk memastikan aplikasi dapat berjalan di server secara normal.

3.2.8. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan dokumentasi pada setiap tahapan tugas akhir mulai dari studi literatur hingga uji coba aplikasi. Penyusunan laporan dikerjakan selama penelitian berlangsung dan juga sebagai aktivitas akhir dalam kegiatan Tugas Akhir ini.

BAB IV
PERANCANGAN

Pada bab ini akan membahas mengenai perancangan dari luaran tugas akhir ini.

4.1. Perancangan Data Scrapping

Yang perlu dipersiapkan sebelum melakukan proses *scrapping* perlu dilakukan pemilihan metode dan *library* yang akan digunakan.

Library Python yang digunakan dalam melakukan *scrapping* untuk Facebook dan Instagram adalah sebagai berikut:

- **Requests**: digunakan untuk melakukan request ke url API atau web, serta mengubah data yang didapat menjadi json
- **PyMySQL**: digunakan untuk menyambungkan serta mengirimkan *query* ke MySQL.
- **Python Dateutil**: digunakan untuk mengubah dan membuat format waktu untuk data yang berbentuk waktu dan tanggal dalam data.
- **ConfigParser**: digunakan untuk memanajemen konfigurasi khusus untuk aplikasi ini.

Selain itu diperlukan juga basis data yang cocok untuk menyimpan data yang telah diambil. Basis data yang digunakan adalah MySQL untuk data dari Facebook maupun Instagram.

Metode yang digunakan peneliti untuk melakukan *scrapping* Facebook adalah melalui API Facebook. Berikut adalah daftar akun media partner yang peneliti pilih dari media sosial Facebook:

- eventsurabaya
- gmoevent2015

Sedangkan untuk Instagram melalui halaman web Instagram langsung. Berikut adalah daftar akun media partner yang peneliti pilih dari media sosial Instagram:

- eventsurabaya
- info_surabaya
- acarasurabaya
- eventpemudasurabaya

4.1.1. Desain Basis Data

Berikut adalah desain tabel untuk menyimpan data hasil scrapping menggunakan basis data MySQL, yang ditunjukkan pada Tabel 4.1.

Tabel 4.1 Desain tabel untuk hasil data scrapping

Atribute	Jenis	Penjelasan	Batasan
created_time	DATETIME	Berisi kapan kiriman tersebut dikirim ke platform	NOT NULL
source	VARCHAR(50)	Berisi sumber dari platform mana data didapat, fb atau ig	NOT NULL, INDEX
user	VARCHAR(50)	Berisi nama user yang mengirimkan kiriman tersebut	NOT NULL
message	TEXT	Berisi teks pesan dari kiriman tersebut	NOT NULL
id	VARCHAR(50)	Berisi id yang dari kiriman tersebut	NOT NULL, INDEX
image_url	VARCHAR(50)	Berisi alamat url dari gambar kiriman, kadang ada kiriman yang tidak memiliki gambar	
source_url	VARCHAR(50)	Berisi alamat url dari kiriman tersebut	NOT NULL
timestamp	TIMESTAMP	Waktu kapan data tersebut diambil dari platform	DEFAULT CURRENT_TIMESTAMP

4.2. Perancangan Data Preprocessing

Preprocessing yang diperlukan pada penelitian ini adalah tokenisasi, dan pemberian label. Data teks yang didapatkan dari proses sebelumnya akan diubah menjadi struktur data yang dapat langsung di-*training* dan di-*test*. Basis data yang digunakan untuk menyimpan adalah MongoDB.

Pengolahan data menggunakan *library* sebagai berikut:

- **NLTK**: digunakan untuk melakukan tokenisasi kata serta pelatihan model.
- **PyMongo**: digunakan untuk menyambungkan serta mengirimkan query ke MongoDB.
- **Flask**: digunakan untuk membuat tampilan berbasis web untuk melakukan proses pemberian label pada data.
- **Python Dateutil**: digunakan untuk mengubah dan membuat format waktu untuk data yang berbentuk waktu dan tanggal dalam data.
- **ConfigParser**: digunakan untuk memanajemen konfigurasi khusus untuk aplikasi ini.

4.2.1. Tokenisasi

Tokenisasi dilakukan untuk mengubah satu kesatuan teks menjadi potongan teks. Pada saat proses tokenisasi, pemisahan kata dibatasi spasi dan baris baru. Maka dari itu untuk membedakan antara spasi dan baris baru maka sebelum dilakukan tokenisasi maka setiap baris baru diubah menjadi “ | ”.

Setelah melakukan teks menjadi token kemudian membuat kumpulan label sebanyak token yang dibuat. Label awal yang diberikan untuk setiap token menggunakan “O”.

4.2.2. Pemberian Label Data

Pemberian label dilakukan menggunakan aplikasi berbasis web. Pembuatan aplikasi dibuat menggunakan kerangka kerja Flask. Sebelum dilakukan pemberian label, data yang didapat dari

scrapping diubah menjadi bentuk yang dapat dilihat pada Kode 4.1. Data yang diubah disimpan di basis data MongoBD.

```
# entries = [
#            {
#                '_id':1,
#                'text' : [ 'lorem', 'ipsum', 'dolor' ],
#                'label': [ 'O', 'B-TIME', 'O'],
#                'timestamp': 2018-03-29 15:11:09.13000,
#                'type': 'bazaar'
#            },
#            {
#                '_id':2,
#                'text' : [
#                       'lorem', ',', 'ipsum', 'dolor'
#                       ],
#                'label': [
#                       'O', 'O', 'B-PLACE', 'I-PLACE'
#                       ],
#                'timestamp': 2018-05-16 10:40:10.67400,
#                'type': 'pendidikan'
#            },
#            {
#                '_id': 3,
#                'text' : [
#                       'lore', 'ipsu', 'dolo', 'sit', '?'
#                       ],
#                'label': [
#                       'B-NAME', 'B-TIME', 'O', 'O', 'O'
#                       ],
#                'timestamp': 2018-05-16 10:41:48.99900,
#            }
#        ]
```

Kode 4.1 Contoh data yang siap diberi label

4.3. Perancangan Information Extractor

Perancangan model menggunakan library sebagai berikut:

- **Sklearn CRFSuite**: digunakan sebagai *library* utama dalam melakukan pelatihan model yang digunakan pada penelitian ini.
- **NLTK**: digunakan untuk membantu pelatihan model dari *library* CRFSuite.
- **PyMongo**: digunakan untuk mengambil data yang akan digunakan sebagai testing dan training.

Sebelum diproses oleh CRFSuite, data diubah menjadi berpasangan antara label dan teks. Maka diperlukan pengubahan sebelum dan setelah data diproses dari basis data ke *library*.

Perubahan ini akan terus digunakan untuk komponen-komponen lain yang ada pada tahap selanjutnya. Maka pada tahap ini dibuat modul Data Adapter, yang akan dimanfaatkan sebagai pengubah data yang disimpan dengan format data yang diperlukan oleh *library*. Hasil akhir dari komponen ini adalah berkas model CRF *Tagger* dan modul data adapter.

4.4. Perancangan Visualisasi Data dan Realtime Scrapper

Pada tahap ini dibuat dua komponen yang akan saling berhubungan. Kedua komponen tersebut memiliki fungsi yang berbeda dan merupakan hasil akhir dari penelitian ini.

Namun, pada saat peneliti menuliskan sub bab 4.4 Instagram melakukan pembaharuan aplikasi yang cakupannya cukup luas sehingga mempengaruhi metode *scrapping* untuk Instagram. Dengan berbagai pertimbangan, maka *realtime scrapper* yang dibuat pada perancangan ini dialihkan ke media sosial Twitter.

4.4.1. Realtime Scrapper

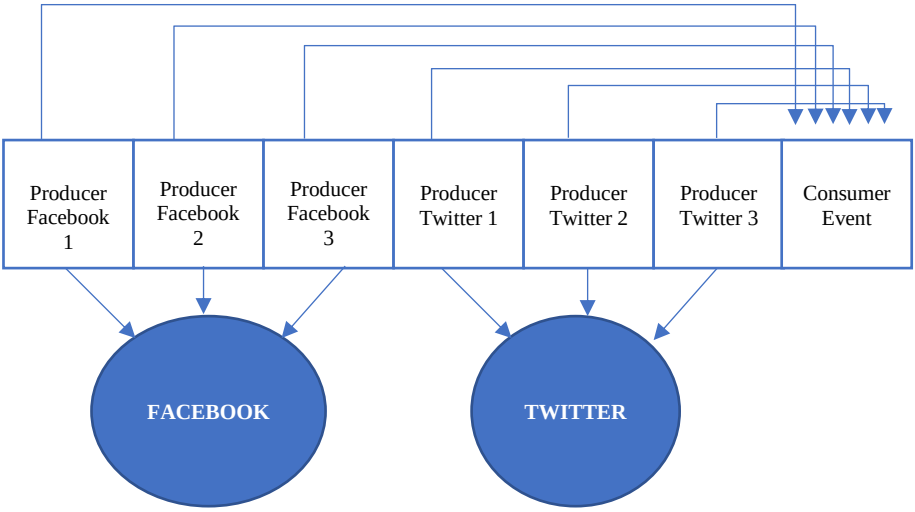
Pada komponen ini memiliki fungsi yang sama dengan komponen scrapper yang dibuat pada perancangan pada sub bab 4.1. Namun pada tahapan ini komponen dibuat dengan konsep *threading* dan *messaging.*

Konsep *threading* dapat diimplementasikan dengan *library* python dengan nama yang sama. Cara menggunakan *library* tersebut membuat anak kelas dari kelas *Thread*.

Sedangkan untuk konsep *messaging* dapat diimplementasikan menggunakan Apache Kafka. Di Python, banyak pilihan *library* Kafka yang dapat dicoba untuk diimplementasikan. Namun, pada penelitian kali ini peneliti hanya memilih PyKafka sebagai

library yang akan menangani hubungan klien pada konsumer dan produser Kafka, tanpa membandingkan lebih lanjut dalam aspek performa dari *library* Python Kafka yang lain.

Dari kedua konsep tersebut peneliti akan membuat thread sebagaimana diilustrasikan pada Gambar 4.1 sebagai berikut:



Gambar 4.1 Ilustrasi hubungan tiap thread pada realtime scrapper

Berikut adalah daftar akun media partner yang peneliti pilih dari media sosial Facebook:

- eventsurabaya
- gmoevent2015

Berikut adalah daftar akun media partner yang peneliti pilih dari media sosial Twitter:

- eventsurabaya
- event_sby

Basis data yang digunakan pada komponen *realtime scrapper* adalah MySQL untuk produser dan MongoDB untuk konsumer.

Pada bagian produser hasil mentah data yang di-s*crapping* disimpan pada struktur tabel yang sama dengan *scrapper* yang telah dibuat sebelumnya sesuai pada Tabel 4.1. Kemudian mengirimkan hasil tersebut ke konsumer.

Pada bagian konsumer akan menerima hasil *scrapping* dari produser dan mengubah teks dari data menjadi label dan token teks menggunakan adapter yang telah dibuat pada komponen sebelumnya dengan struktur yang sama dengan **Kesalahan! S umber referensi tidak ditemukan.**. Label yang diberikan pada teks adalah label yang dihasilkan oleh model yang telah dihasilkan pula pada komponen sebelumnya. Kemudian menyimpan perubahan tersebut ke basis data MongoDB.

Pada komponen *realtime scrapper* diperlukan *library* sebagai berikut:

- **PyKafka**: digunakan sebagai *library* utama dalam hal messaging dari produser ke konsumer.
- **PyMySQL**: digunakan untuk menyimpan data hasil *scrapping* dari produser.
- **PyMongo**: digunakan untuk menyimpan data hasil *tokenisasi* dan pemberian label dari konsumer.
- **Threading**: digunakan agar produser dan konsumer dapat dijalankan secara paralel untuk setiap akun atau halaman yang ada di media sosial.
- **Requests** digunakan untuk melakukan request ke url API atau web, serta mengubah data yang didapat menjadi json khususnya untuk media sosial Facebook.
- **Tweepy**: digunakan khusus untuk melakukan otentifikasi serta *scrapping* data untuk media sosial Twitter.
- **Sklearn CRFSuite**: digunakan sebagai *library* utama dalam melakukan pelabelan dari model yang telah di buat sebelumnya.

- **NLTK**: digunakan untuk melakukan tokenisasi serta membantu melakukan pelabelan model dari *library* CRFSuite.
- **Python Dateutil**: digunakan untuk mengubah dan membuat format waktu untuk data yang berbentuk waktu dan tanggal dalam data.
- **ConfigParser**: digunakan untuk memanajemen konfigurasi khusus untuk aplikasi ini.

4.4.2. Visualisasi Data

Visualisasi data merupakan hasil akhir aplikasi dan dapat diakses oleh masyarakat umum. Berikut ini adalah penjelasan fitur dari komponen visualisasi:

a. Login Akun Google

Fitur login akun diperlukan agar fitur pengingat event dapat diintegrasikan melalui Google Calendar. Login dilakukan menggunakan *sign in button* yang dihubungkan dengan Google API untuk mendapatkan otorisasi kepada pengguna.

b. Halaman Utama

Halaman utama adalah halaman awal setelah pengguna melakukan login ke aplikasi. Pada halaman ini akan menunjukkan halaman peta dengan penunjuk event-event terbaru dari media sosial.

c. Daftar Event

Daftar event merupakan tampilan berupa daftar dengan penjelasan singkat event dengan melampirkan poster serta informasi penting yang telah diolah sebelumnya.

d. Detail Event

Detail event adalah tampilan detail dari satu event beserta peta lokasi tempat kegiatan diadakan.

e. Mencari Event

Pada fitur ini tampilan hampir sama seperti fitur daftar event, namun daftar yang ditampilkan lebih spesifik pada kegiatan yang berhubungan dengan kata kunci tertentu. Pencarian juga

bisa berdasarkan rentang waktu atau lokasi kegiatan bersangkutan.

f. Tambahkan Pengingat Event

Menambahkan pengingat event dapat ditemukan pada detail event berupa tombol. Fitur ini akan menambahkan event terkait ke dalam pengingat Google Calendar.

4.5. Perancangan Uji Coba Aplikasi

Untuk meningkatkan kualitas aplikasi maka diperlukan uji coba terotomatisasi dengan menggunakan uji coba unit testing. Unit testing tidak dilakukan pada semua bagian modul secara keseluruhan namun lebih fokus pada modul penting yang penggunaannya dipakai pada lebih dari satu komponen atau merupakan modul yang sifatnya dapat digunakan dalam pengembangan lebih lanjut (*reuseable*).

Modul yang direncanakan akan dilakukan unit testing adalah sebagai berikut:

4.5.1. Modul MongoDB

Modul ini adalah modul utama yang digunakan dalam menghubungkan klien, mengambil dan memasukkan data ke server. Modul ini digunakan pada komponen *Preprocess*, *Realtime Scrapper*, serta Visualisasi data.

Modul ini dikembangkan dengan memanfaatkan *library* PyMongo dengan menerapkan *Create*, *Read*, *Update*, dan *Delete* secara *Object Oriented Programming* (OOP).

4.5.2. Modul ModelAdapter

Modul ini adalah modul yang digunakan dalam mengubah bentuk data yang disimpan menjadi data yang dapat diproses oleh *library* NLTK dan CRFSuite. Modul ini digunakan pada komponen *Preprocess*, *Realtime Scrapper*, serta Visualisasi data.

4.5.3. Modul Crawl

Modul ini adalah modul yang digunakan untuk melakukan proses *scrapping* secara *realtime*. Modul ini dibuat dengan menerapkan implementasi untuk dua media sosial yaitu Facebook dan Twitter. Modul ini hanya digunakan pada komponen *Realtime Scrapper*, namun peneliti merasa bahwa modul ini masih bisa dikembangkan lebih lanjut dengan menambah implementasi baru dari media sosial lain.

Kemungkinan lain dikemudian hari terdapat perubahan tidak terduga dari media sosial yang dijadikan sasaran (Facebook, Twitter). Hal ini akan menyebabkan perlu dilakukan pembaharuan serta penyesuaian modul dengan kondisi media sosial pada saat itu.

4.5.4. Modul DataProvider

Modul ini adalah modul yang digunakan untuk mengubah format data yang disimpan di basis data MongoDB yang masih berupa token dan label menjadi entitas-entitas yang memiliki makna untuk ditampilkan pada komponen visualisasi data.

Modul ini dirasa memerlukan pembaharuan apabila tampilan aplikasi akan diubah dalam skala besar. Maka pembuatan uji coba diperlukan untuk modul ini.

BAB V
IMPLEMENTASI

Pada bab ini, akan dijelaskan mengenai implementasi dari perancangan yang telah dilakukan sesuai dengan metode pengembangan yang dibuat.

5.1. Persiapan Pembuatan Aplikasi

Setiap komponen yang dibuat menggunakan *virtual environment* pada Python yang bernama pipenv. Pipenv dibuat berdasarkan Pipfile yang berlaku untuk tiap komponen yang berbeda.

Perangkat Lunak yang digunakan dalam pengembangan antara lain adalah sebagai berikut

- Sistem Operasi : Windows 10 Pro
- Bahasa Pemrograman : Python
- Tools :
    - Microsoft Visual Studio Code
    - Google Chrome
    - Postman
    - HeidiSQL
    - MongoDB Compass
    - Putty
    - Apache Kafka

5.2. Pembuatan Komponen ScrapperEvent

Komponen ScraperEvent ini mencakup tahapan untuk mengumpulkan data dari media sosial, serta mengubah data tersebut menjadi data yang sudah tertokenisasi dan memiliki label awal.

5.2.1. Modul Database

Pada komponen ini diperlukan modul yang digunakan untuk memasukkan data ke MongoDB. Modul ini akan digunakan untuk komponen-komponen lain sehingga modul ini akan

dibuat agar dapat mencakup semua kebutuhan yang sesuai pada semua komponen.

Untuk modul ini peneliti akan membuat modul yang dapat diimplementasikan secara modular sehingga jenis basis data yang digunakan tidak hanya MongoDB namun dapat diimplementasikan pula pada jenis basis data NoSQL yang lain. Modul MongoDB berada pada folder database.

Berikut adalah class diagram untuk Modul Database pada Tabel 5.1. Dokumentasi kode program dapat di lihat pada LAMPIRAN A.

Tabel 5.1 Class Diagram MongoDB

MongoDB(AbstractDB)
Self.config : ConfigParser Self.mongo : MongoClient Self.db_name : str Self.dataset : str
__init__(self, config : str, config_name : str, db_name : str, dataset : str) _check_status(self) : void setDataset(self, dataset : str) : void putData(self, data : dict) : void getEntries(self, offset : int, limit : int) : cursor getAll(self) : cursor getId(self, id : str) : cursor getTimestamp(self, id : str) : datetime setTimestamp(self, id : str) : void setData(self, data : dict) : void setType(self, id : str, type : string) : void removeType(self, id : str) : void setDataset(self, dataset : str) : void getDataset(self) : MongoClient

Untuk menggunakan modul MongoDB sebelumnya diperlukan untuk menginisiasi kelas MongoDB dengan menyertakan file konfigurasi yang berisi informasi minimal HOST, dan PORT, atau bila diperlukan USER, dan PASS.

5.2.2. Scrapping Instagram

Dalam membangun sub komponen untuk melakukan *scrapping* Instagram maka peneliti memutuskan untuk membuat satu kelas dengan class diagram pada Tabel 5.2. Dokumentasi kode program dapat dilihat pada LAMPIRAN A

Tabel 5.2 Class Diagram Instagram

Instagram
Self.config : ConfigParser Self.base_url : str Self.page_ids : list Self.limit : int
__init__(self, configFile : str, limit : int, pageIds : list) connect(self) : void disconnect(self) : void countData(self) : int getRequest(self, request : str, loop : int, page_id : str) : void insert(self, current_post : object, page_id : str) : void start(self) : void

Scrapping Data melalui instagram dilakukan dengan mengambil nilai JSON yang diambil dari alamat tautan dengan akhiran __a=1. Metode ini tidak perlu menyertakan client id dan client secret dan sebagainya karena metode ini dapat dilakukan tanpa otorisasi seperti layaknya API pada umumnya. Metode ini berlaku hingga bulan Februari 2018.

5.2.3. Scrapping Facebook

Dalam membangun sub komponen untuk melakukan *scrapping* Facebook maka peneliti memutuskan untuk membuat satu kelas dengan *class diagram* pada Tabel 5.3 sebagai berikut:

Tabel 5.3 Class Diagram Facebook

Facebook
Self.config : ConfigParser Self.client_id : str

Self.client_secret : str Self.version : str Self.base_url : str Self.page_ids : list Self.limit : int Self.request_token : str Self.token : object Self.access_token : str
__init__(self, configFile : str, limit : int, pageIds : list) connect(self) : void disconnect(self) : void getRequest(self, request : str, page_id : str) : object insert(self, current_post : object, page_id : str) : void start(self) : void

Sebelum menggunakan modul ini pastikan sudah mempersiapkan client ID, client secret, dan versi API dengan cara mendaftarkan aplikasi ke Facebook Developer. Scrapping data dari Facebook dilakukan melalui API. Untuk setiap referensi API Facebook yang tersedia pada suatu versi akan berbeda dengan versi yang lain. Versi yang digunakan pada modul adalah versi 2.11.

5.2.4. Pembuatan Data Tagging

Dalam mengubah struktur data yang di simpan di MySQL menjadi data yang bisa digunakan pada komponen berikutnya, maka peneliti membuat suatu *class* dalam menangani salah satu tahapan tersebut, yang disebut sebagai *class* Tagging.

Agar pelabelan dilakukan menggunakan *dataset* yang seimbang maka peneliti membuat batasan kiriman yang akan diambil untuk setiap sumber serta pengguna yang berbeda pada tahap awal. Namun untuk selanjutnya data dapat ditambah apabila diperlukan.

Class Tagging yang digunakan terdiri dari fungsi seperti yang ditunjukkan pada *class diagram* pada Tabel 5.4. Dokumentasi kode program dapat di lihat pada LAMPIRAN A.

Tabel 5.4 Class Diagram Tagging

Tagging
Self.config : ConfigParser Self.client : MongoClient Self.conn : Connection Self.cursor : DictCursor
__init__(self, configFile : str, table_name : str) connect(self) : void disconnect(self) : void countdata(self) : list countdataset(self, version : int) : list createdataset(self, limit : int) : void createtag(self, name : str) : void tokenize(self, data : dict, dataset_name : str, reset : boolean) : void newdata(self, limit : int, version : int) : void

Class Tagging ini digunakan untuk membuat dataset dari hasil scrapping yang disimpan di MySQL menjadi data yang siap untuk dilabel yang disimpan pada MongoDB. Cara menggunakan kelas ini adalah yang pertama membuat dataset dengan fungsi createdataset(self, limit : int). Fungsi ini akan membuat tabel daftar data yang akan dibuat dataset. Saat menjalankan fungsi ini maka daftar dataset akan direset kembali dan dibuat mulai dari versi 0. Setelah membuat dataset, kemudian membuat *tag*, atau membuat label dengan fungsi createtag(self, name : str). Fungsi ini berfungsi dalam mengubah data yang terdaftar menjadi token dan memasukkan ke database MongoDB. Untuk menambahkan dataset baru bisa menggunakan fungsi newdata(self, limit : int, version : int). Pada fungsi ini daftar dataset bisa diperbarui kemudian saat menjalankan fungsi createtag sekali lagi maka akan memasukkan data dengan versi terbaru pada database MongoDB.

5.3. Pembuatan Komponen TaggingEvent

Komponen TaggingEvent ini berguna sebagai aplikasi pemberi label yang akan dilakukan oleh beberapa orang yang disebut dengan *Tagger*. Komponen ini berbasis web dan dapat diakses

dalam jaringan. Maka dari itu aplikasi serta basis data juga perlu di-*hosting* secara daring pula.

5.3.1. Tautan

Berikut adalah daftar tautan yang dapat diakses pada aplikasi:

a. Halaman utama

Berikut adalah tautan menuju halaman utama yang diperlihatkan pada Kode 5.1.

```
GET
/
/page/{int:page}
/page/{int:page}/limit/{int:limit}
```

Kode 5.1 Tautan untuk halaman utama

Pada halaman yang ditunjukkan pada Kode 5.1 terdapat tampilan utama yang dapat diakses oleh anotator yang terdiri dari beberapa halaman yang berisi sejumlah potongan data serta ID dari data tersebut. Pada halaman ini juga terdapat tombol guideline yang akan menampilkan aturan yang perlu dipahami anotator sebelum melakukan pelabelan pada aplikasi.

Pada halaman utama terdapat dua masukkan yang dapat diberikan. Yang pertama adalah nomor halaman, kemudian yang kedua adalah batas jumlah data yang ditampilkan pada satu halaman.

b. Halaman Uji Coba Model

Berikut adalah tautan menuju halaman uji coba model yang diperlihatkan pada Kode 5.2.

```
GET
/test
/page/{int:page}/test
/page/{int:page}/limit/{int:limit}/test
```

Kode 5.2 Tautan untuk halaman uji coba model

Pada halaman yang ditunjukkan pada Kode 5.2 terdapat tampilan yang hampir sama dengan halaman utama. Namun, pada halaman ini terdapat bagian tambahan di bawah tiap data yang menunjukkan hasil prediksi dari model jika model sudah dibuat sebelumnya. Halaman ini berfungsi sebagai analisis secara manual untuk melihat ciri-ciri serta keakuratan dari model dilihat dari setiap data yang diprediksi.

5.3.2. API Endpoints

Untuk mendapatkan aplikasi yang ramah pengguna maka diperlukan tampilan yang dapat mengubah data secara *asyncronize* dengan menggunakan metode AJAX. Dalam melakukan AJAX maka diperlukan juga API endpoint sebagai titik pengambilan data serta manipulasi data

Berikut adalah daftar API endpoint yang dapat diakses oleh aplikasi:

a. Mengubah data label

Mengubah data label dilakukan saat menekan salah satu token kemudian memilih label. Saat memilih label maka *javascript* akan mengirimkan permintaan ke server dengan metode POST atau PUT. Referensi API untuk mengubah data label yang dapat dilihat pada Kode 5.3.

PUT POST /api/{string:id}/{int:index}		
Field		
parameter	description	type
tag	The new label that changed from the token in index	string
Response		
{"_id":"10928084098","index":7,"tag":"B-NAME"}		

Kode 5.3 Referensi API untuk mengubah data label

Referensi API ini membutuhkan masukkan berupa ID data dan indeks dari token tersebut. Untuk *request body* berupa JSON

dengan kunci parameter “tag”. Tag bisa disebut dengan nama label. Hasil kembalian dari API juga tipe data JSON. Nilai kembalian berupa _id, indeks, dan tag.

b. Mengambil waktu pengubahan terakhir

Berikut adalah referensi API untuk waktu pengubahan terakhir yang dapat dilihat pada Kode 5.4.

```
GET
/api/timestamp/{string:id}
```

Response

```
{"_id":"10928084098","timestamp":"2013-03-01T00:00:00Z"}
```

Kode 5.4 Referensi API untuk mengambil waktu pengubahan terakhir

Referensi API ini membutuhkan masukkan berupa ID data. Nilai kembalian berupa _id, dan timestamp.

c. Menambahkan kategori data

Berikut adalah referensi API untuk menambahkan kategori data yang dapat dilihat pada Kode 5.5.

```
POST
/api/type/{string:id}/{string:categoy}
```

Response

```
{"success":true}
```

Kode 5.5 Referensi API untuk menambahkan kategori data

Referensi API ini membutuhkan masukkan berupa ID data dan kategori dari data tersebut. Nilai kembalian berupa status berhasil.

d. Menghapus kategori data

Berikut adalah referensi API untuk menghapus kategori data yang dapat dilihat pada Kode 5.6.

```
POST
/api/type/{string:id}/
```

Response

```
{"success":true}
```

Kode 5.6 Referensi API untuk menghapus kategori data

Referensi API ini membutuhkan masukkan berupa ID data dan harus dikirim menggunakan metode POST. Nilai kembalian berupa status berhasil.

5.4. Pembuatan Komponen ModelEvent

Komponen ModelEvent ini mencakup tahapan pelatihan model dan pengujian model. Secara garis besar data yang sudah dilabeli pada komponen sebelumnya akan dilatih menggunakan *library* NLTK. Namun, ditemukan bahwa NLTK tidak mampu melakukan pengujian pada model yang dibuat sehingga peneliti memutuskan untuk membuat pengujian secara manual.

Pada tahapan ini diperlukan suatu modul yang dapat mengubah format data menjadi agar dapat diproses oleh *library*. Namun data yang diubah itu harus dapat diubah kembali menjadi format awal agar dapat disimpan ke basis data. Modul yang akan dibuat untuk mengakomodasi fungsi-fungsi tersebut adalah Modul Adapter.

Pada Modul Adapter harapannya dapat mengakomodasi pula fungsi yang diperlukan oleh komponen pada tahap berikutnya. Maka dari itu model hasil pelatihan juga akan diimplementasikan pula pada modul ini.

5.4.1. Pelatihan Model

Pelatihan model dilakukan dengan menggunakan menggunakan modul adapter dan modul database. Berikut adalah *pseudocode* yang digunakan dalam pelatihan model pada Kode 5.7.

```
GET data yang sudah terlabel
FOR data:
    Mengubah format data
Membuat model menggunakan data
Menguji model
```

Kode 5.7 Pseudocode pelatihan model

5.4.2. Modul Model Adapter

Pada modul adapter hanya diperlukan satu class saja. Pada modul ini dibuat class dengan nama DataAdapter. Berikut ini adalah class diagram dari class DataAdapter pada Tabel 5.5. Dokumentasi kode Program dapat dilihat pada LAMPIRAN A.

Tabel 5.5 Class Diagram DataAdapter

DataAdapter
Self.tagger : CRFTagger Self.data_tagging : list Self.data_testing : list
__init__(self, data : list/pymongo.cursor.Cursor) count(self, data : list/pymongo.cursor.Cursor) : int set_data(self, data : list) : void tokenize_tag(self, text : str) : list, list for_tagging_testing(self, data : list) : list, list for_testing(self, data : list) : list for_tagging(self, data : list) : list tag(self, data : list) : list labeling(self) : list evaluating(self) : list training(self, data : list) : void

Modul ini digunakan untuk mengubah format data agar sesuai dengan format library. Selain itu modul ini digunakan dalam proses pelatihan dan pengujian pada model.

5.4.3. Pengujian Model

Pengujian model dilakukan menggunakan modul adapter dan modul database seperti halnya dengan pelatihan model. Pengujian dilakukan menggunakan fungsi yang ada di modul adapter dengan *psedudocode* yang ditunjukkan pada Kode 5.8.

```
FOREACH data in datas:
   Hasil_model
   FOREACH label_training in data:
       Label_model = Hasil_model[indeks]
       Jumlah_per_label++
       Jumlah_semua++
       IF label == Label_model
           Jumlah_benar_per_label++
```

```
            Jumlah_benar_semua++
        IF label_training == nama_label
            Jumlah_label_training++
        IF label_model == nama_label
            Jumlah_label_model++
Akurasi = Jumlah_benar_semua / Jumlah_semua
Recall = Jumlah_benar_per_label / Jumlah_label_model
Presisi = Jumlah_benar_per_label / Jumlah_label_training
```

Kode 5.8 Pseudocode Uji Coba Model

Pengujian dilakukan dengan menghitung akurasi, presisi dan *recall* untuk lima kategori INFO, NAME, PLACE, TIME, dan OTHER. Dari kelima kategori tersebut akan dicari akurasi, presisi dan *recall* secara keseluruhan, namun pada perhitungan mengabaikan kategori OTHER karena peran label ini tidak terlalu signifikan dan memiliki jumlah yang terlalu banyak pula.

5.5. Pembuatan Komponen KafkaEvent

Komponen KafkaEvent berguna sebagai pengatur metode pengambilan data secara *realtime* melalui Apache Kafka. Komponen ini terdiri dari dua berkas utama yaitu berkas producer.py dan consumer.py.

5.5.1. Produser

Pada berkas producer.py diperlukan *class* yang mengimplementasikan *Thread* sebagaimana *class* tersebut dapat dipanggil sebagai objek dan dapat dijalankan secara paralel. Permintaan ke API dibuat dengan default delay 60 detik. Namun karena batas permintaan perhari untuk API Facebook maka dilakukan permintaan setiap 5 jam.

5.5.2. Modul Crawl

Modul ini memiliki fungsi untuk mengatur data yang yang diambil dari sosial media. Class Crawl ini memiliki dua subclass yang mengimplementasikan untuk media sosial yang berbeda, yaitu CrawlFacebook dan CrawlTwitter. Dokumentasi kode program dapat dilihat pada LAMPIRAN A.

Berikut adalah class diagram dari class Crawl pada Tabel 5.6.

Tabel 5.6 Class Diagram Crawl

Crawl inherit threading.Thread
Self.client : KafkaClient Self.topic : str Self.producer : Producer Self.name : str Self.delay : int
__init__(self, client : KafkaClient, topic : str, name : str, producer : Producer, delay : int) get_producer(self, topic : str, sync : boolean, use_rdkafka : boolean, delivery_reports : boolean, max_request_size : int, **kwargs) : Producer stop(self) : void run(self) : void run_onetime(self) : void run_realtime(self) : void producing(self, value : str, key : str) : void serializer(self, value : str, pk : str) : bytes, bytes connect(self) : void disconnect(self) : void getNewestID(self, source : str, page_id : str) : cursor remove(self, id : str) : void

Berikut adalah class diagram dari class CrawlFacebook pada Tabel 5.7.

Tabel 5.7 Class Diagram CrawlFacebook

CrawlFacebook inherit Crawl
Self.client : KafkaClient Self.topic : str Self.account : str
__init__(self, client : KafkaClient, topic : str, **kwargs) run(self) : void run_onetime(self) : void getRequest(self, request : str, page_id : str, since_id : int) : object getNewestID(self, page_id : str) : int insert(self, current_post : object, page_id : str) : void

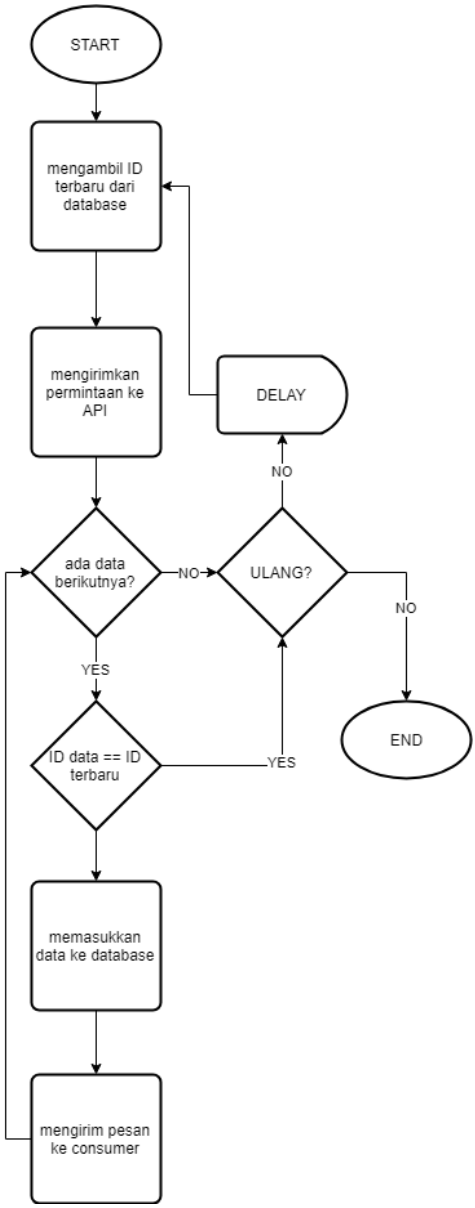
Berikut adalah class diagram dari class CrawlTwitter pada Tabel 5.8.

Tabel 5.8 Class Diagram CrawlTwitter

CrawlTwitter inherit Crawl
Self.client : KafkaClient Self.topic : str Self.account : str
__init__(self, client : KafkaClient, topic : str, **kwargs) run(self) : void run_onetime(self) : void getRequest(self, page_id : str, since_id : int, max_id : int) : object getNewestID(self, page_id : str) : int insert(self, current_post : object, page_id : str) : void

Setiap kelas mengimplementasikan cara melakukan permintaan API pada masing-masing sosial media. Maka dari itu untuk proses yang sama diimplementasikan langsung pada kelas Crawl sebagai *super class*. Jangka waktu antara permintaan satu dengan permintaaan berikutnya secara default diatur untuk 60 detik, namun untuk API Facebook yang memiliki batasan permintaan per hari maka dilakukan pengaturan jangka waktu yang lebih lama. Diatur untuk jangka waktu 5 jam.

Berikut adalah diagram alir dari proses yang berjalan pada modul Crawl yang ditunjukkan pada Gambar 5.1.

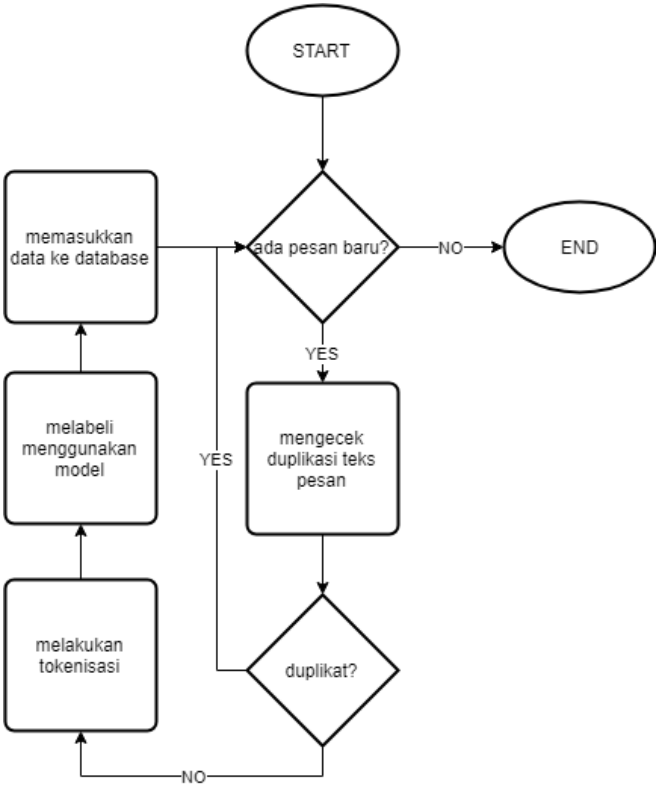


Gambar 5.1 Diagram alir pada proses yang dijalankan producer

5.5.3. Konsumer

Pada berkas consumer.py diperlukan *class* untuk mengubah format struktur data dari teks pesan menjadi token kemudian dilakukan pelabelan pada token tersebut menggunakan model yang telah dibuat pada komponen sebelumnya. Class yang melakukan hal tersebut adalah class DataAdapter seperti yang diperlihatkan pada Tabel 5.5.

Berikut adalah diagram alir untuk proses yang berjalan pada consumer yang ditunjukkan pada



Gambar 5.2 Diagram alir proses pada consumer

5.5.4. Modul Data Provider

Modul ini digunakan sebagai menambahkan data mentah dari hasil pengambilan data *realtime* yang disimpan di MongoDB menjadi data yang dapat ditampilkan dan query secara bebas untuk Komponen ExtractEvent. Modul ini bergantung pada Modul Database karena modul ini akan mengambil dan mengubah data yang tersimpan di MongoDB.

Berikut adalah class diagram untuk Modul Data Provider pada Tabel 5.9. Dokumentasi kode program dapat dilihat pada LAMPIRAN A.

Tabel 5.9 Class Diagram DataProvider

DataProvider(object)
Self.md : MongoDB
__init__(self, config : str, mapClient : googlemaps.Client, mongo : MongoDB) getData(self, id : str, save : boolean) : dict extractText(self, labels : list, text : list) : list removeDuplicate(self, old_list : list, key : str) : list parserEvent(self, label : str, array : list, created : datetime) : str/list/datetime time_process(self, text : str, created : datetime) : datetime place_process(self, text : str) : list

Modul data provider dipanggil oleh consumer untuk melakukan pelabelan pada token. Proses yang dilakukan adalah dengan mengambil data berdasarkan ID. Setelah data didapatkan kemudian dilakukan pengambilan entitas dari token yang telah dilabeli tadi. Entitas yang didapat kemudian diubah menjadi tipe data yang sesuai, misal untuk label PLACE dilakukan pencarian lokasi berdasarkan teks untuk mendapatkan informasi lokasi secara lengkap yang dilakukan menggunakan Google Maps API. Sedangkan untuk label TIME dilakukan pengubahan menjadi tipe data datetime sehingga dapat

diurutkan berdasarkan waktu. Untuk label lain seperti NAME dan INFO diubah menjadi bentuk kata utuh yang disusun dari token-token yang ada pada entitas tersebut.

5.6. Pembuatan Komponen ExtractEvent

Komponen ExtractEvent adalah hasil akhir dari penelitian ini. Komponen ini terdiri dari aplikasi berbasis web dengan Flask serta modul Data Provider.

5.6.1. Tautan

Berikut adalah daftar tautan yang dapat diakses pada aplikasi:

a. Sign in

Berikut adalah tautan *sign in* ke aplikasi yang diperlihatkan pada Kode 5.9.

```
GET
/signin
```

Kode 5.9 Tautan sign in

Sign in digunakan untuk mendapatkan akses ke akun google. Pada aplikasi ini diperlukan akun google untuk mengakses google kalender saat membuat pengingat untuk pengguna. Setelah pengguna *login*, halaman akan mengarahkan ke tautan yang sebelumnya dikunjungi.

b. Log out

Berikut adalah tautan untuk *logout* dari aplikasi yang diperlihatkan pada Kode 5.10.

```
GET
/logout
```

Kode 5.10 Tautan untuk logout

Logout digunakan untuk mengeluarkan akun google dari aplikasi.

c. Buat pengingat

Berikut adalah tautan membuat pengingat yang diperlihatkan pada Kode 5.11.

```
GET
/add_reminder/{string:id}
```

Kode 5.11 Tautan untuk membuat pengingat

Menambahkan pengingat dapat mengakses tautan dengan ID *event* bersangkutan. Namun, membuat pengingat hanya berlaku untuk kegiatan yang memiliki entitas waktu pada detail *event* tersebut.

Fitur membuat pengingat dibuat dengan meminta otoritas untuk mengubah kalender google pada akun pengguna. Setelah itu, aplikasi dapat mengirimkan permintaan untuk menambahkan event baru pada kalender melalui API Google Calendar.

d. Halaman utama

Berikut adalah tautan menuju halaman utama yang diperlihatkan pada Kode 5.12.

```
GET
/
/search/{string:query}
```

Kode 5.12 Tautan untuk halaman utama

Pada halaman utama, menampilan peta dengan label yang menunjukkan pernah atau akan diadakan *event* di tempat tersebut. Dengan menekan tombol “show events” maka daftar *event* akan muncul. Selain itu bisa juga melakukan pencarian berdasarkan tempat kemudian dapat mengecek apakah tempat tersebut diadakan suatu kegiatan atau tidak.

Diperlukan masukkan pada tautan halaman utama jika ingin melakukan pencarian nama tempat. Pencarian dapat dilakukan dengan menuliskan kata kunci nama tempat sehingga penanda akan muncul di peta pada lokasi yang sesuai dengan kata kunci yang dimasukkan.

e. Halaman detail event

Berikut adalah tautan menuju halaman detail event yang diperlihatkan pada Kode 5.13.

```
GET
/detail/{string:id}
```

Kode 5.13 Tautan untuk detail event

Pada halaman detail event tampilan event hampir sama dengan saat melihat daftar event yang membedakan adalah tombol Buat Pengingat. Apabila keadaan belum *login* maka tombol tersebut akan mengarahkan pada tautan *login*. Selain itu, apabila tidak ditemukan entitas waktu pada *event* tersebut maka tombol tidak akan ditampilkan pula.

Pada tautan detail event diperlukan ID dari data. Apabila ID data tidak valid maka halaman akan menunjukkan pesan error "404 Halaman tidak ditemukan".

f. Halaman featured

Berikut adalah tautan menuju halaman *featured* yang diperlihatkan pada Kode 5.14.

```
GET
/featured
/featured/page/{int:page}
/featured/page/{int:page}/limit/{int:limit}
```

Kode 5.14 Tautan untuk halaman featured

Halaman ini digunakan untuk menampilkan daftar event yang diurutkan berdasarkan waktu pengiriman di sosial media. Pada halaman featured terdapat dua masukkan yang dapat diberikan. Yang pertama adalah nomor halaman, kemudian yang kedua adalah batas jumlah data yang ditampilkan pada satu halaman.

g. Halaman upcoming

Berikut adalah tautan menuju halaman *upcoming* yang diperlihatkan pada Kode 5.15.

```
GET
/upcoming
```

Kode 5.15 Tautan untuk halaman upcoming

Halaman ini digunakan untuk menampilkan peta dengan dengan lokasi yang memiliki event yang akan datang. Tampilan halaman ini memiliki kesamaan dengan halaman utama namun yang membedakan adalah jumlah lokasi yang ditampilkan lebih sedikit.

5.6.2. API Endpoints

Untuk mendapatkan aplikasi yang ramah pengguna maka diperlukan tampilan yang dapat mengubah data secara *asyncronize* dengan menggunakan metode AJAX. Dalam melakukan AJAX maka diperlukan juga API endpoint sebagai titik pengambilan data serta manipulasi data

Berikut adalah daftar API endpoint yang dapat diakses oleh aplikasi:

a. API Place

Berikut adalah tautan untuk API Place yang diperlihatkan pada Kode 5.16.

POST /api/place		
Field		
parameter	description	type
name	Name of place	string
address	Full address	string
Location	Geometry location based on latitude and langitude {lat: -7.2646499, lng: 112.7407103}	object
Response		

```
HTML Page with event data inside.
```

Kode 5.16 Referensi API untuk mencari event di suatu lokasi

API *endpoint* untuk mengambil event berdasarkan detail lokasi yang didapatkan dari google places API. Referensi API ini membutuhkan masukkan berupa ID data dan indeks dari token tersebut. Untuk *request body* berupa JSON dengan kunci parameter *name*, *address*, dan *location*. Ketiga parameter tersebut adalah parameter yang didapatkan dari API Google Place untuk melakukan pencarian tempat. Untuk *name* dan *address* memiliki tipe data string sedangkan untuk *location* merupakan object dengan dua parameter didalamnya berupa lat dan lng. Respon yang diberikan oleh API ini berupa HTML yang siap dipasang pada *sidebar* yang ada pada aplikasi

5.7. Pembuatan Unit Testing

Dalam melakukan *unit testing* maka diperlukan dokumen *testing* agar uji coba dapat lebih mudah dibuat atau dimodifikasi untuk pengembangan aplikasi lebih lanjut. Berikut adalah dokumen unit testing untuk setiap modul.

5.7.1. Modul MongoDB

Berikut adalah daftar skenario dan test case untuk modul MongoDB yang ditunjukkan pada Tabel 5.10.

Tabel 5.10 Scenario dan Test Case Modul MongoDB

No.	Scenario	No.	Test Case
101	Mengambil data menggunakan modul MongoDB	001	Mengambil satu data dengan ID spesifik
102	Memasukkan data ke database menggunakan modul MongoDB	001	Menginput data dan mengecek kembali dari database
		002	Menginputkan data lebih dari sekali dengan ID yang sama
		003	Menginputkan data dan mengambil data menggunakan limit dan offset
103	Mengubah data dari database menggunakan modul MongoDB	001	Mengubah data label dengan data ID dan indeks tertentu
		002	Mengubah data label dengan data ID yang sesuai namun input lain salah
		003	Mengubah data label dengan input yang salah

		004	Mengubah data pada type
		005	Mengubah data pada type kemudian menghapus kembali
		006	Mengubah data pada type dengan menggunakan ID yang salah
104	Menghapus data dengan teks yang sama menggunakan modul MongoDB	001	Memasukkan banyak data kemudian menghapus yang memiliki full_text yang sama

Dari Tabel 5.10, akan dijabarkan lebih detail untuk setiap Test Case untuk Skenario pada Tabel 5.11 hingga Tabel 5.21.

Tabel 5.11 Test Case mengambil satu data dengan ID spesifik

No.	: 101
Scenario	: Mengambil data dengan menggunakan modul MongoDB
File Target	: database/md.py
Last Run	: 02/07/18 22:26
Duration	: 5,67 detik
Test Case No.	: 001
Test Name	: test.md_test.GetMongoTestCase.test_get_id
Test Case Description	: Mengambil satu data dengan ID spesifik

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	id masukkan	nilai _id pada variable result	☑
2	AssertEqual	mockup daftar label	nilai label pada variable result	☑
3	AssertEqual	mockup daftar text	nilai text pada variable result	☑
4	AssertEqual	mockup waktu dibuat dengan tipe data datetime	nilai created time pada variable result	☑
5	AssertEqual	mockup url sumber	nilai source_url pada variable result	☑
6	AssertEqual	mockup url gambar	nilai image_url pada variable result	☑

Tabel 5.12 Test Case menginput data dan mengecek kembali dari database

No.	: 102
Scenario	: Memasukkan data ke database menggunakan modul MongoDB
File Target	: database/md.py
Last Run	: 03/07/18 03:53
Duration	: 9,21 detik
Test Case No.	: 001
Test Name	: test.md_test.PutMongoTestCase.test_put_data_0
Test Case Description	: Menginput data dan mengecek kembali dari database

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	data yang telah di input ke database	data yang diambil menggunakan input id	☑
2	AssertEqual	value dengan tipe datetime dari timestamp pada data yang telah diinput	timestamp yang diambil menggunakan input id	☑

Tabel 5.13 Test Case menginputkan data lebih dari sekali dengan ID yang sama

No.	: 102
Scenario	: Memasukkan data ke database menggunakan modul MongoDB
File Target	: database/md.py
Last Run	: 03/07/18 22:26
Duration	: 9,21 detik
Test Case No.	: 002
Test Name	: test.md_test.PutMongoTestCase.test_put_data_1
Test Case Description	: Menginput data lebih dari sekali dengan ID yang sama

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	data pertama yang di input ke database	data yang diambil menggunakan input id	☑
2	AssertEqual	data yang diambil menggunakan input id	data yang diambil menggunakan input id	☑
3	AssertEqual	jumlah data pada database hanya 1	jumlah semua data yang diambil dari database	☑

Tabel 5.14 Test Case menginput data dan mengambil data menggunakan limit dan offset

No.	: 102			
Scenario	: Memasukkan data ke database menggunakan modul MongoDB			
File Target	: database/md.py			
Last Run	: 03/07/18 04:11			
Duration	: 5,63 detik			
Test Case No.	: 003			
Test Name	: test.md_test.PutMongoTestCase.test_put_data_2			
Test Case Description	: Menginput data dan mengambil data menggunakan limit dan offset			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	jumlah data yang terambil adalah 2	jumlah data yang diambil menggunakan offset 1 dan limit 2	☑
2	AssertEqual	data input pada indeks 1 sampai indeks 3	bentuk list dari hasul data yang diambil menggunakan offset 1 dan limit 2	☑

Tabel 5.15 Test Case mengubah data label dengan data ID dan indeks tertentu

No.	: 103			
Scenario	: Mengubah data dari database menggunakan modul MongoDB			
File Target	: database/md.py			
Last Run	: 03/07/18 04:16			
Duration	: 5,90 detik			
Test Case No.	: 001			
Test Name	: test.md_test.SetMongoTestCase.test_set_data_0			
Test Case Description	: Mengubah data label dengan data ID dan indeks tertentu			
Assertions				
No	Assertion	Expected	Actual	Result

1	AssertEqual	data label yang di inputkan	value label dari data yang diambil pada data indeks yang diinputkan	☑

Tabel 5.16 Test Case mengubah data label dengan data ID yang sesuai namun input lain salah

No.	: 103			
Scenario	: Mengubah data dari database menggunakan modul MongoDB			
File Target	: database/md.py			
Last Run	: 03/07/18 04:20			
Duration	: 4,96 detik			
Test Case No.	: 002			
Test Name	: test.md_test.SetMongoTestCase.test_set_data_1			
Test Case Description	: Mengubah data label dengan data ID yang sesuai namun input lain salah			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertRaises	KeyError	saat melakukan pengubahan data	☑

Tabel 5.17 Test Case mengubah data label dengan input yang salah

No.	: 103			
Scenario	: Mengubah data dari database menggunakan modul MongoDB			
File Target	: database/md.py			
Last Run	: 03/07/18 04:20			
Duration	: 4,96 detik			
Test Case No.	: 003			
Test Name	: test.md_test.SetMongoTestCase.test_set_data_2			
Test Case Description	: Mengubah data label dengan input yang salah			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertRaises	KeyError	saat melakukan pengubahan data	☑

Tabel 5.18 Test Case mengubah tipe data

No.	: 103

Scenario		: Mengubah data dari database menggunakan modul MongoDB		
File Target		: database/md.py		
Last Run		: 03/07/18 04:24		
Duration		: 5,59 detik		
Test Case No.		: 004		
Test Name		: test.md_test.SetMongoTestCase.test_set_type_0		
Test Case Description		: Mengubah data pada type		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	tipe yang diinputkan	value tipe pada data yang diambil	☑

Tabel 5.19 Test Case mengubah data type kemudian menghapus kembali

No.		: 103		
Scenario		: Mengubah data dari database menggunakan modul MongoDB		
File Target		: database/md.py		
Last Run		: 03/07/18 04:20		
Duration		: 4,96 detik		
Test Case No.		: 005		
Test Name		: test.md_test.SetMongoTestCase.test_set_type_1		
Test Case Description		: Mengubah data pada type kemudian menghapus kembali		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	tipe yang diinputkan	value tipe pada data yang diambil	☑
2	AssertRaises	KeyError	saat melakukan mengambil value type dari data yang diambil	☑

Tabel 5.20 Test Case mengubah data type dengan ID yang salah

No.	: 103
Scenario	: Mengubah data dari database menggunakan modul MongoDB
File Target	: database/md.py
Last Run	: 03/07/18 04:32
Duration	: 5,51 detik
Test Case No.	: 006

Test Name		: test.md_test.SetMongoTestCase.test_set_type_2		
Test Case Description		: Mengubah data pada type dengan menggunakan ID yang salah		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	None	data yang diambil menggunakan id	☑

Tabel 5.21 Test Case memasukkan banyak data kemudian menghapus yang memiliki message yang sama

No.		: 104		
Scenario		: Menghapus data dengan teks yang sama menggunakan modul MongoDB		
File Target		: database/md.py		
Last Run		: 03/07/18 05:04		
Duration		: 6,58 detik		
Test Case No.		: 001		
Test Name		: test.md_test.RemoveDuplicateTestCase.test_remove		
Test Case Description		: Memasukkan banyak data kemudian menghapus yang memiliki full_text yang sama		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	jumlah data yang diinputkan ke database adalah 5	jumlah semua data yang diambil dari database	☑
2	AssertEqual	jumlah data yang ada di database adalah 1 setelah dilakukan penghapusan	jumlah semua data yang diambil dari database	☑

5.7.2. Modul Model Adapter

Berikut adalah daftar skenario dan test case untuk modul Model Adapter yang ditunjukkan pada Tabel 5.22.

Tabel 5.22 Scenario dan Test Case Modul Model Adapter

No.	Scenario	No.	Test Case
301	Menggunakan modul Model Adapter tanpa inisiasi data	001	Membuat data untuk dilabel oleh model
		002	Membuat data untuk uji coba pada model

		003	Melakukan tokenisasi pada teks
		004	Melakukan tokenisasi pada teks multi baris
		005	Melakukan pelabelan pada data
302	Menggunakan modul Model Adapter dengan inisiasi data	001	Membuat data untuk dilabel oleh model
		002	Membuat data untuk uji coba pada model
		003	Melakukan pelabelan pada data

Dari Tabel 5.22, akan dijabarkan lebih detail untuk setiap Test Case untuk Skenario pada Tabel 5.23 hingga Tabel 5.30.

Tabel 5.23 Test Case membuat data untuk dilabel oleh model

No.	: 301			
Scenario	: Menggunakan modul Model Adapter tanpa inisiasi data			
File Target	: model/adapter.py			
Last Run	: 03/07/18 19:58			
Duration	: 0,02 detik			
Test Case No.	: 001			
Test Name	: test.adapter_test.AdapterTestCase.test_for_tagging_0			
Test Case Description	: Membuat data untuk dilabel oleh model			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data mockup yang terdiri dari daftar kumpulan token	Hasil pengubahan data yang ada pada database menjadi kumpulan token	☑

Tabel 5.24 Test Case membuat data untuk uji coba pada model

No.	: 301
Scenario	: Menggunakan modul Model Adapter tanpa inisiasi data
File Target	: model/adapter.py
Last Run	: 03/07/18 20:00
Duration	: 0,01 detik

Test Case No.	: 002			
Test Name	: test.adapter_test.AdapterTestCase.test_for_testing_0			
Test Case Description	: Membuat data untuk uji coba pada model			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data mockup yang terdiri dari daftar kumpulan token beserta labelnya	Hasil pengubahan data yang ada pada database menjadi kumpulan token beserta labelnya	☑

Tabel 5.25 Test Case melakukan tokenisasi pada teks

No.	: 301			
Scenario	: Menggunakan modul Model Adapter tanpa inisiasi data			
File Target	: model/adapter.py			
Last Run	: 03/07/18 20:07			
Duration	: 0,01 detik			
Test Case No.	: 003			
Test Name	: test.adapter_test. AdapterTestCase.test_for_tokenize_tag_0			
Test Case Description	: Melakukan tokenisasi pada teks			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data mockup kumpulan token	Hasil kembalian pertama pada pengubahan teks menjadi kumpulan token	☑
2.	AssertEqual	Jumlah data pada mockup kumpulan label	Hasil kembalian kedua pada pengubahan teks menjadi kumpulan token	☑

Tabel 5.26 Test Case melakukan tokenisasi pada teks multi baris

No.	: 301
Scenario	: Menggunakan modul Model Adapter tanpa inisiasi data
File Target	: model/adapter.py

Last Run	: 03/07/18 20:07			
Duration	: 0,01 detik			
Test Case No.	: 004			
Test Name	: test.adapter_test. AdapterTestCase.test_for_tokenize_tag_1			
Test Case Description	: Melakukan tokenisasi pada teks multi baris			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data mockup kumpulan token	Hasil kembalian pertama pada pengubahan teks menjadi kumpulan token	☑
2.	AssertEqual	Jumlah data pada mockup kumpulan label	Hasil kembalian kedua pada pengubahan teks menjadi kumpulan token	☑

Tabel 5.27 Test Case melakukan pelabelan pada data

No.	: 301			
Scenario	: Menggunakan modul Model Adapter tanpa inisiasi data			
File Target	: model/adapter.py			
Last Run	: 03/07/18 20:11			
Duration	: 0,01 detik			
Test Case No.	: 005			
Test Name	: test.adapter_test.AdapterTestCase.test_for_tag_0			
Test Case Description	: Melakukan pelabelan pada data			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data kumpulan token yang diiterasi	Hasil data yang telah dilabel diiterasi kemudian diambil value pada indeks ke 0	☑

Tabel 5.28 Test Case membuat data untuk dilabel oleh model

No.	: 302
Scenario	: Menggunakan modul Model Adapter dengan inisiasi data
File Target	: model/adapter.py

Last Run	: 03/07/18 20:41			
Duration	: 0,01 detik			
Test Case No.	: 001			
Test Name	: test.adapter_test. AdapterWithInputTestCase.test_for_tagging_1			
Test Case Description	: Membuat data untuk dilabel oleh model			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data mockup yang terdiri dari daftar kumpulan token	Hasil pengubahan data yang ada pada database menjadi kumpulan token	☑

Tabel 5.29 Test Case membuat data untuk uji coba pada model

No.	: 302			
Scenario	: Menggunakan modul Model Adapter dengan inisiasi data			
File Target	: model/adapter.py			
Last Run	: 03/07/18 20:41			
Duration	: 0,01 detik			
Test Case No.	: 002			
Test Name	: test.adapter_test. AdapterWithInputTestCase.test_for_testing_1			
Test Case Description	: Membuat data untuk uji coba pada model			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data mockup yang terdiri dari daftar kumpulan token beserta labelnya	Hasil pengubahan data yang ada pada database menjadi kumpulan token beserta labelnya	☑

Tabel 5.30 Test Case melakukan pelabelan pada data

No.	: 302
Scenario	: Menggunakan modul Model Adapter dengan inisiasi data
File Target	: model/adapter.py
Last Run	: 03/07/18 20:43
Duration	: 0,01 detik

Test Case No.	: 003			
Test Name	: test.adapter_test. AdapterWithInputTestCase.test_for_labeling_0			
Test Case Description	: Melakukan pelabelan pada data			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data kumpulan token yang diiterasi	Hasil data yang telah dilabel diiterasi kemudian diambil value pada indeks ke 0	☑
1	AssertEqual	Data kumpulan label yang diiterasi	Hasil data yang telah dilabel diiterasi kemudian diambil value pada indeks ke 1	☑

5.7.3. Modul Crawl

Berikut adalah daftar skenario dan test case untuk modul Crawl yang ditunjukkan pada Tabel 5.31Tabel 5.34.

Tabel 5.31 Scenario dan Test Case Modul Crawl

No.	Scenario	No.	Test Case
401	Melakukan pengambilan data menggunakan modul Crawl	001	Mengambil ID dari data terbaru
		002	Mengambil ID dari data terbaru dengan input yang salah

Dari Tabel 5.31, akan dijabarkan lebih detail untuk setiap Test Case untuk Skenario pada Tabel 5.32 dan Tabel 5.33.

Tabel 5.32 Test Case mengambil ID dari data terbaru

No.	: 401
Scenario	: Melakukan pengambilan data menggunakan modul Crawl
File Target	: crawl.py
Last Run	: 04/07/18 15:12
Duration	: 2,41 detik
Test Case No.	: 001
Test Name	: test.crawl_test.CrawlTestCase.test_get_newest_id

Test Case Description : Mengambil ID dari data terbaru				
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Tipe data integer	Tipe data hasil kembalian dari mengambil id terbaru dari data facebook	☑
2	AssertEqual	Tipe data integer	Tipe data hasil kembalian dari mengambil id terbaru dari data twitter	☑

Tabel 5.33 Test Case mengambil ID dari data terbaru dengan input yang salah

No.	: 401			
Scenario	: Melakukan pengambilan data menggunakan modul Crawl			
File Target	: crawl.py			
Last Run	: 04/07/18 15:12			
Duration	: 0,92 detik			
Test Case No.	: 002			
Test Name	: test.crawl_test.CrawlTestCase.test_get_newest_id			
Test Case Description	: Mengambil ID dari data terbaru dengan input yang salah			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Tipe data integer	Tipe data hasil kembalian dari mengambil id terbaru dari data facebook	☑
2	AssertEqual	Tipe data integer	Tipe data hasil kembalian dari mengambil id terbaru dari data twitter	☑

5.7.4. Modul Data Provider

Berikut adalah daftar skenario dan test case untuk modul Data Provider yang ditunjukkan pada Tabel 5.34.

Tabel 5.34 Scenario dan Test Case Modul Data Provider

No.	Scenario	No.	Test Case
201	Melakukan pengolahan pada data waktu	001	Mengubah teks menjadi waktu dengan tipe datetime, format tanggal bulan tahun
		002	Mengubah teks menjadi waktu dengan tipe datetime, format bulan tanggal, tahun
		003	Mengubah teks menjadi waktu dengan tipe datetime, format tanggal-bulan-tahun
202	Melakukan pengolahan menggunakan Google Maps Client	001	Mengubah teks menjadi lokasi dengan format nama, alamat dan posisi geometri
		002	Mengambil data dengan ID dari database untuk dilakukan pengolahan teks
203	Melakukan pengolahan kumpulan token dan label menjadi entitas	001	Mengolah kumpulan token dan label dengan format B1 B2 I1 O O
		002	Mengolah kumpulan token dan label dengan format B1 I1 O B2 B3 O O
		003	Mengolah kumpulan token dan label dengan format B1 B1 O B2 I3 O O
		004	Mengolah kumpulan token dan label dengan format B1 I1 O B1 I1 O O dengan teks yang sama

Dari Tabel 5.34, akan dijabarkan lebih detail untuk setiap Test Case untuk Skenario pada Tabel 5.35 hingga Tabel 5.43.

Tabel 5.35 Test Case mengubah teks menjadi waktu dengan tipe datetime, format tanggal bulan tahun

No.	: 201
Scenario	: Melakukan pengolahan pada data waktu
File Target	: data/provider.py
Last Run	: 03/07/18 09:31
Duration	: 2,41 detik
Test Case No.	: 001

Test Name		: test.provider_test.TimeTestCase.test_time_process_0		
Test Case Description		: Mengubah teks menjadi waktu dengan tipe datetime, format tanggal bulan tahun		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data dengan tipe datetime	Hasil pengolahan waktu dengan input teks	☑

Tabel 5.36 Test Case mengubah teks menjadi waktu dengan tipe datetime, format tanggal-bulan-tahun

No.		: 201		
Scenario		: Melakukan pengolahan pada data waktu		
File Target		: data/provider.py		
Last Run		: 03/07/18 09:34		
Duration		: 1,48 detik		
Test Case No.		: 003		
Test Name		: test.provider_test.TimeTestCase.test_time_process_2		
Test Case Description		: Mengubah teks menjadi waktu dengan tipe datetime, format tanggal-bulan-tahun		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data dengan tipe datetime	Hasil pengolahan waktu dengan input teks	☑

Tabel 5.37 Test Case mengubah teks menjadi waktu dengan tipe datetime, format bulan tanggal, tahun

No.		: 201		
Scenario		: Melakukan pengolahan pada data waktu		
File Target		: data/provider.py		
Last Run		: 03/07/18 09:32		
Duration		: 1,72 detik		
Test Case No.		: 002		
Test Name		: test.provider_test.TimeTestCase.test_time_process_1		
Test Case Description		: Mengubah teks menjadi waktu dengan tipe datetime, format bulan tanggal, tahun		
Assertions				
No	Assertion	Expected	Actual	Result

1	AssertEqual	Data dengan tipe datetime	Hasil pengolahan waktu dengan input teks	☑

Tabel 5.38 Test Case mengambil data dengan ID dari database untuk dilakukan pengolahan teks

No.	: 202
Scenario	: Melakukan pengolahan menggunakan Google Maps Client
File Target	: data/provider.py
Last Run	: 03/07/18 09:22
Duration	: 1,59 detik
Test Case No.	: 001
Test Name	: test.provider_test.MapsTestCase.test_place_process
Test Case Description	: Mengambil data dengan ID dari database untuk dilakukan pengolahan teks

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	mockup data dengan nama alamat dan posisi geometri yang sudah dicari sebelumnya	Hasil pengolahan tempat dengan input teks	☑

Tabel 5.39 Test Case mengubah teks menjadi lokasi dengan format nama, alamat dan posisi geometri

No.	: 202
Scenario	: Melakukan pengolahan menggunakan Google Maps Client
File Target	: data/provider.py
Last Run	: 03/07/18 09:26
Duration	: 8,12 detik
Test Case No.	: 002
Test Name	: test.provider_test.MapsTestCase.test_get_data
Test Case Description	: Mengubah teks menjadi lokasi dengan format nama, alamat dan posisi geometri

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	mockup data yang ada di database dan hasil olahannya	Hasil data yang diambil dan	☑

			diolah dengan input id	

Tabel 5.40 Test Case mengolah kumpulan token dan label dengan format B1 B2 I1 O O

No.	: 203			
Scenario	: Melakukan pengolahan kumpulan token dan label menjadi entitas			
File Target	: data/provider.py			
Last Run	: 03/07/18 09:02			
Duration	: 1,31 detik			
Test Case No.	: 001			
Test Name	: test.provider_test. ExtractTextTestCase.test_extract_text_0			
Test Case Description	: Mengolah kumpulan token dan label dengan format B1 B2 I1 O O			
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Kumpulan data label dengan PLACE dan TIME	Hasil data yang diproses dari kumpulan teks dan kumpulan label dengan urutan B-PLACE, B-TIME, I-TIME, O, O	☑

Tabel 5.41 Test Case mengolah kumpulan token dan label dengan format B1 I1 O B2 B3 O O

No.	: 203
Scenario	: Melakukan pengolahan kumpulan token dan label menjadi entitas
File Target	: data/provider.py
Last Run	: 03/07/18 09:06
Duration	: 0,86 detik
Test Case No.	: 002
Test Name	: test.provider_test. ExtractTextTestCase.test_extract_text_1
Test Case Description	: Mengolah kumpulan token dan label dengan format B1 I1 O B2 B3 O O
Assertions	

No	Assertion	Expected	Actual	Result
1	AssertEqual	Kumpulan data label dengan NAME, PLACE dan TIME	Hasil data yang diproses dari kumpulan teks dan kumpulan label dengan urutan B-NAME, I-NAME, O, B-TIME, B-PLACE, O, O	☑

Tabel 5.42 Test Case mengolah kumpulan token dan label dengan format B1 B1 O B2 I3 O O

No.	: 203
Scenario	: Melakukan pengolahan kumpulan token dan label menjadi entitas
File Target	: data/provider.py
Last Run	: 03/07/18 09:09
Duration	: 1,14 detik
Test Case No.	: 003
Test Name	: test.provider_test. ExtractTextTestCase.test_extract_text_2
Test Case Description	: Mengolah kumpulan token dan label dengan format B1 B1 O B2 I3 O O

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	Kumpulan data label dengan INFO, INFO dan TIME	Hasil data yang diproses dari kumpulan teks dan kumpulan label dengan urutan B-INFO, B-INFO, O, B-TIME, I-PLACE, O, O	☑

Tabel 5.43 Test Case mengolah kumpulan token dan label dengan format B1 I1 O B1 I1 O O dengan teks yang sama

No.	: 203
Scenario	: Melakukan pengolahan kumpulan token dan label menjadi entitas

File Target	: data/provider.py
Last Run	: 03/07/18 09:06
Duration	: 1,14 detik
Test Case No.	: 004
Test Name	: test.provider_test. ExtractTextTestCase.test_extract_text_3
Test Case Description	: Mengolah kumpulan token dan label dengan format B1 I1 O B1 I1 O O dengan teks yang sama

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	Kumpulan data label dengan INFO	Hasil data yang diproses dari kumpulan teks dan kumpulan label dengan urutan B-INFO, B-INFO, O, B-INFO, I-INFO, O, O	☑

Halaman ini sengaja dikosongkan

BAB VI
HASIL DAN PEMBAHASAN

6.1. Hasil Data *Scrapping*

Setelah dilakukan *scrapping*, dilakukan pemilahan dari enam sumber user dari sosial media Facebook dan Instagram. Maka jumlah data yang terlalu banyak hanya dipakai sebagian agar dataset lebih imbang. Pemilihan data dilakukan dengan cara acak. Sehingga didapatkan data dengan perincian sebagai berikut pada Tabel 6.1.

Tabel 6.1 Perincian jumlah data scrapping

Username	Sosial Media	Jumlah Data Dimiliki	Jumlah Data yang tidak Terduplikasi
acarasurabaya	Instagram	282	282
eventpemudasurabaya	Instagram	4914	1225
eventsurabaya	Instagram	12008	5974
info_surabaya	Instagram	2053	1946
gmoevent2015	Facebook	65	63
eventsurabaya	Facebook	4920	2244
		24242	**11734**

Berikut adalah sampel data *scrapping* pada Tabel 6.2.

Tabel 6.2 Sampel Data Scrapping

created_time	source	user	message	id	image_url	source_url	timestamp

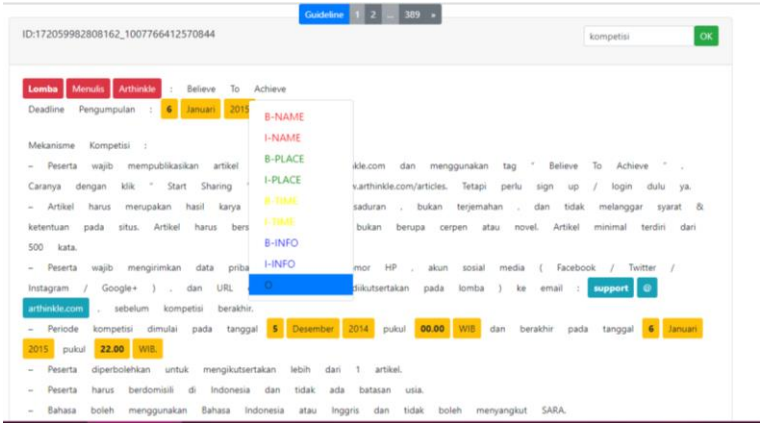
28/05/2018 21:47	fb	eventsurabaya	Seminar "7 Keys to Succesfull Family Business" Pelaksanaan : Hari Kamis, 31 Mei 2018 (09.00 - 13.00 WIB) Tempat : "International Room" IBMT School of Management Alamat : Jalan Raya Kupang Baru No.8 Keynote : Imam Wijoyo (Chairman of IBMT School of Management) Pembicara : Agus Sanetyo (Action Coach dan Owner Mie Mapan Surabaya) Fasilitas : E-Certificate Informasi: Handy (087853134449)	1720599828081622_2154364924577648	https://scontent.xx.fbcdn.net/v/t1.0-9/33870831_2154364931244314_3099117692053880832_n.jpg?_nc_cat=0&oh=a1b1e170c3280eb1ca0ffe475d6bd36e&oe=5BC40432	https://www.facebook.com/eventsurabaya/photos/a.223848010962692.72684.1720599828081622/2154364924577648/?type=3	28/05/2018 22:45
28/05/2018 20:52	fb	eventsurabaya	JAWA TIMUR ! IKUTILAH KOMPETISI BERGENGSI "PTC GOT TALENT 2018" Lomba semua bakat: MC, presenter, dance, nyanyi, modeling, sulap, story telling, musik, drum atau bakat keren lainnya !! Daftar segera early bird hanya 150.000/kategori/jenis. Normal price: 200K Daftarkan diri Anda! WA: 081938211402 (fast respon) Line@ : Sekolahmcsurabaya (follow using @) IG: sekolahmcsurabaya	1720599828081622_2154304874583653		https://www.facebook.com/eventsurabaya/photos/a.223848010962692.72684.1720599828081622/2154304787916995/?type=3	28/05/2018 22:45
28/05/2018 20:17	fb	eventsurabaya	Beauty Class with Dokter Vanda Flawless Make Up Look (Self Make Up) Minggu, 5 Agustus 2018 Sky Ballroom Hotel Fave Mex Surabaya Daftar: WA 081232324196 WA 081232582426	1720599828081622_2154272851253522	https://scontent.xx.fbcdn.net/v/t1.0-9/33780773_2154272854586855_4904388756909326336_n.jpg?_nc_cat=0&oh=e1c503703d3f3192659dea71140d643c&oe=5BC50D53	https://www.facebook.com/eventsurabaya/photos/a.223848010962692.72684.1720599828081622/2154272851253522/?type=3	28/05/2018 22:45
25/05/2018 13:05	tw	eventsurabaya	Tarungkan bakatmu! Festival Nggenjreng Akustikan Bersama Royke	9999997885307160000	http://pbs.twimg.com/media/DdIU	https://twitter.com/events	28/05/2018 22:37

			“Arek Band” | HTM: 50rb | RSVP: 08155229554 https://t.co/dqLweEhh23		ifxVMAE0Y3V.jpg	urabaya/status/995881062860386304/photo/1	
25/05/2018 13:03	tw	events urabaya	Yukk buat siswa SMP, SMA, dan mahasiswa sederajat tunjukkan kreatifitasmu melalui Short Film Competition ITEM 2018! Total hadiah 11JUTA++ menantikamu! Yukk ikutan, kami tunggu karyamu! https://t.co/DmxEzBMpuG	99999928195295000	http://pbs.twimg.com/media/Dd3da1mUQAAKjBg.jpg	https://twitter.com/eventsurabaya/status/999191530552565761/photo/1	28/05/2018 22:37
25/05/2018 13:00	tw	events urabaya	Mari kita umat kristen bersatu untuk berdoa bagi kesejahteraan Indonesia dalam acara Jesus Reigns! PARADE, PRAISE, PRAY. Jumat, 1 Juni 2018 at The Square Ballroom (Basuki Rahmat 16-18) | IG JesusReignsIndonesia / WA: 083875757595 https://t.co/1QtS43tkXW	99999863395626000	http://pbs.twimg.com/media/DdtJDS6UQAA6uDr.jpg	https://twitter.com/eventsurabaya/status/998472243055943169/photo/1	28/05/2018 22:37

Dilihat dari hasil *scrapping* bahwa tidak semua kiriman memiliki gambar. Serta pesan yang dikirim tidak selalu berhubungan dengan *event*.

6.2. Hasil Data *Tagging*

Aplikasi yang digunakan untuk melakukan pelabelan data dapat dilihat pada Gambar 6.1 dan kode program lebih lengkap dapat dilihat di tautan berikut https://github.com/dewzzjr/tagging-event.



Gambar 6.1 Tampilan aplikasi tagging

Berikut adalah sampel data setelah dilakukan pelabelan pada Tabel 6.3.

Tabel 6.3 Sampel Data Tagging

```
{'_id': '172059982808162_1007766412570844',
 'label': ['B-NAME', 'I-NAME', 'I-NAME', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'B-TIME', 'I-TIME', 'I-TIME', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'B-INFO', 'I-INFO', 'I-INFO', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'B-TIME', 'I-TIME', 'I-TIME', 'O',
           'B-TIME', 'I-TIME', 'O', 'O', 'O', 'O', 'B-TIME', 'I-TIME', 'I-TIME',
           'O', 'B-TIME', 'I-TIME', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'B-INFO', 'O', 'B-INFO',
           'O', 'O', 'O', 'O', 'O'],
 'text': ['Lomba', 'Menulis', 'Arthinkle', ':', 'Believe',
          'To', 'Achieve', '|', 'Deadline', 'Pengumpulan',
          ':', '6', 'Januari', '2015', '|', '|', 'Mekanisme',
          'Kompetisi', ':', '|', '', 'Peserta', 'wajib',
          'mempublikasikan', 'artikel', 'di', 'situs',
          'www.arthinkle.com', 'dan', 'menggunakan', 'tag',
          '', 'Believe', 'To', 'Achieve', '', '.',
          'Caranya', 'dengan', 'klik', '', 'Start',
          'Sharing', '', 'di', 'halaman',
```

```
          'www.arthinkle.com/articles.', 'Tetapi',
          'perlu', 'sign', 'up', '/', 'login', 'dulu',
          'ya.', '|', '', 'Artikel', 'harus', 'merupakan',
          'hasil', 'karya', 'sendiri', ',', 'bukan', 'saduran',
          ',', 'bukan', 'terjemahan', ',', 'dan',
          'tidak', 'melanggar', 'syarat', '&', 'ketentuan',
          'pada', 'situs.', 'Artikel', 'harus', 'bersifat',
          'inspiratif', 'dan', 'bukan', 'berupa', 'cerpen',
          'atau', 'novel.', 'Artikel', 'minimal', 'terdiri',
          'dari', '500', 'kata.', '|', '', 'Peserta', 'wajib',
          'mengirimkan', 'data', 'pribadi', '(', 'nama', ',',
          'nomor', 'HP', ',', 'akun', 'sosial', 'media',
          '(', 'Facebook', '/', 'Twitter', '/', 'Instagram',
          '/', 'Google+', ')', ',', 'dan', 'URL', 'dari',
          'artikel', 'yang', 'diikutsertakan', 'pada', 'lomba',
          ')', 'ke', 'email', ':', 'support', '@', 'arthinkle.com',
          ',', 'sebelum', 'kompetisi', 'berakhir.', '|', '',
          'Periode', 'kompetisi', 'dimulai', 'pada', 'tanggal',
          '5', 'Desember', '2014', 'pukul', '00.00', 'WIB',
          'dan', 'berakhir', 'pada', 'tanggal', '6', 'Januari',
          '2015', 'pukul', '22.00', 'WIB.', '|', '', 'Peserta',
          'diperbolehkan', 'untuk', 'mengikutsertakan', 'lebih',
          'dari', '1', 'artikel.', '|', '', 'Peserta', 'harus',
          'berdomisili', 'di', 'Indonesia', 'dan', 'tidak',
          'ada', 'batasan', 'usia.', '|', '', 'Bahasa',
          'boleh', 'menggunakan', 'Bahasa', 'Indonesia', 'atau',
          'Inggris', 'dan', 'tidak', 'boleh', 'menyangkut', 'SARA.',
          '|', '|', 'Hadiah', ':', '|', '', 'Juara', '1', ':',
          'Rp.1.000.000', '|', '', 'Juara', '2', ':',
          'Rp.500.000', '|', '', 'Juara', '3', ':',
          'Rp.300.000', '|', '|', 'More', 'Information',
          ':', '|', 'Bit.ly/1yqZddh', '|', 'www.arthinkle.com',
          '|', '|', 'http', ':',
          '//eventsurabaya.net/lomba-menulis-arthinkle-believe-to-achieve/'],
 'timestamp': datetime.datetime(2018, 5, 23, 14, 5, 18, 254000),
 'type': 'kompetisi'}
```

```
{'_id': '172059982808162_1011137912233694',
 'label': ['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'B-NAME', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'B-INFO', 'I-INFO', 'O', 'B-INFO',
           'I-INFO', 'O', 'B-INFO', 'I-INFO', 'I-INFO', 'I-INFO', 'I-INFO'],
 'text': ['Udah', 'nyoba', 'Cheese', 'Cake', 'yg', 'bisa', 'dibawa', 'Hang',
          'Out', '?', 'Zni_cakeinabox', 'Paling', 'PAS', 'buat', 'Kamu',
          ',', 'Keep', 'Healthy', '&', 'Yummy', '|', '|', 'Contact', 'Person',
          ':', '|', 'BBM', '29C6509B', '|', 'Whatsapp', '081234513071',
          '|', 'Twitter', '/', 'IG', ':', 'zni_cakeinabox'],
 'timestamp': datetime.datetime(2018, 5, 3, 0, 6, 42, 894000)}
```

```
{'_id': '172059982808162_103741973071497',
 'label': ['O', 'O', 'O'],
 'text': ['http', ':', '//www.eventsurabaya.net/pet-exhibition-2/'],
 'timestamp': datetime.datetime(2018, 3, 29, 15, 11, 6, 902000)}
```

```
{'_id': '172059982808162_1043243319023153',
 'label': ['B-NAME', 'I-NAME', 'I-NAME', 'I-NAME', 'O', 'O', 'O', 'O', 'B-TIME',
           'I-TIME', 'I-TIME', 'I-TIME', 'I-TIME', 'O', 'O', 'B-PLACE', 'I-
PLACE',
           'I-PLACE', 'I-PLACE', 'I-PLACE', 'O', 'B-TIME', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O'],
 'text': ['Charity', 'Life', 'Run', 'Surabaya', '(', 'CLRS', ')',
          '|', 'Minggu', ',', '15', 'Februari', '2015', '|', 'At',
          'Parkir', 'Timur', 'Plaza', '', 'Surabaya', '|', '5am',
          'till', 'drop', '|', '|', 'Featuring', ':', '|', '',
          'Karen', 'Garrett', '|', '', 'Winky', 'Wiryawan', '|',
          '', 'Eddie', 'Tripleks', '|', '|', 'http', ':',
          '//eventsurabaya.net/charity-life-run-surabaya-clrs/'],
```

```
  'timestamp': datetime.datetime(2018, 5, 15, 10, 7, 44, 346000)}
```

```
{'_id': '172059982808162_1044926398854845',
 'label': ['B-NAME', 'B-NAME', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',
           'O', 'O', 'O', 'O', 'O', 'O', 'O'],
 'text': ['Planopolis', '2015', ':', 'Coastal', 'Planning', '',
          'Lomba', 'Karya', 'Tulis', 'Ilmiah', 'Mahasiswa',
          'Tingkat', 'Nasional', '', '|', 'Tema', ':',
          'Perencanaan', 'Pesisir', 'Berkelanjutan', '|',
          'Sub', 'Tema', ':', '|', '-', 'Reklamasi', 'Pantai',
          '|', '-', 'Minapolitan', '|', '-', 'Pengelolaan',
          'Sumber', 'Daya', 'Pesisir', '|', '-', 'Waterfront',
          'City', '|', '-', 'Kebijakan', 'Ekonomi', 'Pesisir',
          '|', '-', 'Mitigasi', 'Bencana', 'Pesisir', '|',
          '|', 'http', ':',
          '//eventsurabaya.net/planopolis-2015-coastal-planning-lomba-karya-
tulis-ilmiah-mahasiswa-tingkat-nasional/'],
 'timestamp': datetime.datetime(2018, 3, 29, 15, 11, 6, 909000)}
```

6.3. Hasil Model CRF

Berikut adalah hasil dari model CRF yang dibuat dengan menggunakan *library* NLTK:

6.3.1. Model NLTK tanpa modifikasi

Setelah dilakukan pelabelan didapatkan sejumlah 1164 data yang telah dilabel, diketahui jumlah label dengan prediksi benar berjumlah 129190. Sedangkan jumlah total label adalah 136732, maka didapatkan akurasi dengan perhitungan yang dapat dilihat pada Persamaan 6.1.

$$AKURASI = \frac{TRUE\ POSITIF}{TOTAL\ LABEL} = \frac{129190}{136732} = 94{,}48\%$$

Persamaan 6.1 Perhitungan akurasi model

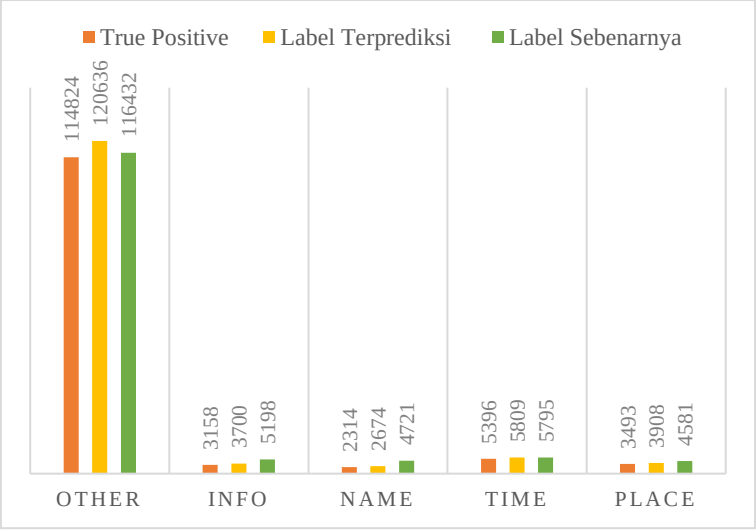
Perincian data untuk setiap label dapat dilihat pada Tabel 6.4.

Tabel 6.4 Rekap jumlah label hasil uji coba

Label	True Positive	Label Terprediksi	Label Sebenarnya
OTHER	114824	120636	116432

INFO	3158	3700	5198
NAME	2314	2674	4721
TIME	5396	5809	5795
PLACE	3493	3908	4581

Berikut ini adalah visualisasi dari Tabel 6.4 dapat dilihat pada Gambar 6.2.



Gambar 6.2 Diagram rekap label hasil uji coba

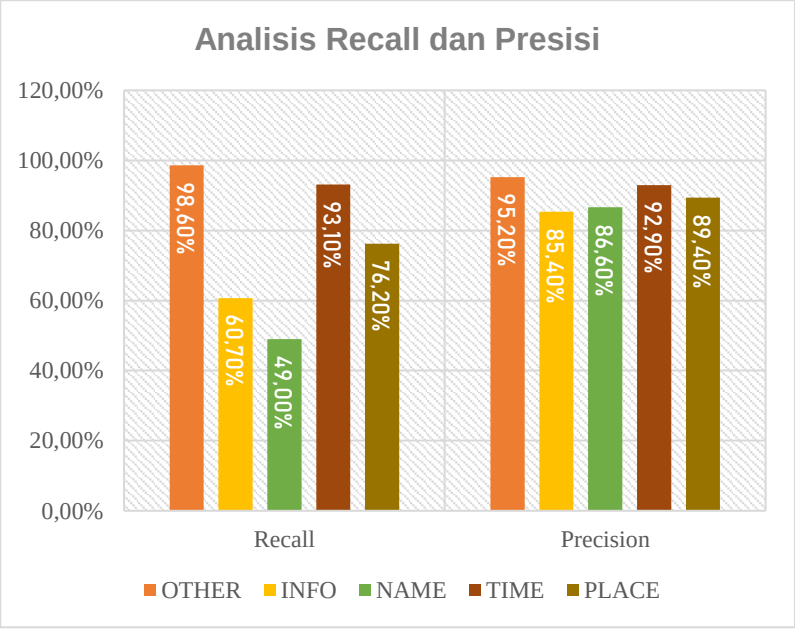
Dari rincian tersebut dapat dilihat bahwa jumlah label OTHER sangat mendominasi dari semua data yang ada, hal ini menyebabkan nilai akurasi yang diberikan didominasi oleh data OTHER. Namun dilihat perbandingan antara label terprediksi dan label sebenarnya label TIME dan OTHER memiliki jumlah yang hampir seimbang. Sedangkan pada PLACE, NAME, dan INFO masih terjadi ketidakseimbangan antara label terprediksi dengan label sebenarnya sehingga menyebabkan jumlah label yang diprediksi dengan benar semakin sedikit pula. Perbedaan yang signifikan terlihat pada label NAME dan INFO dengan perbandingan 2674 dengan 4721, dan 3700 dengan 5198.

Dari perincian tersebut didapatkan *recall* dan presisi sebagai berikut pada Tabel 6.5.

Tabel 6.5 Rekap recall dan presisi per label

Label	Recall	Precision
OTHER	98,6%	95,2%
INFO	60,7%	85,4%
NAME	49,0%	86,6%
TIME	93,1%	92,9%
PLACE	76,2%	89,4%
Rata-rata	75,5%	89,9%

Berikut ini adalah visualisasi dari Tabel 6.5 dapat dilihat pada Gambar 6.3.



Gambar 6.3 Diagram analisis recall dan presisi

Pada diagram di atas presisi model untuk kelima label memiliki nilai yang cukup tinggi dan stabil sehingga mendapatkan rata-rata 89,9%. Namun kekurangan dari model yang dibuat adalah pada aspek *recall* yang rendah terutama pada label INFO dan NAME, dengan nilai 60,7% dan 49,0%. Berbeda dengan label TIME dan OTHER yang memiliki *recall* dan presisi yang hampir mendekati sempurna. Rata-rata *recall* semua label 75,5%.

6.3.2. Model NLTK dengan modifikasi

Setelah mendapatkan hasil yang kurang memuaskan dari model yang didapatkan dari penggunaan NLTK secara default. Maka peneliti memutuskan untuk melakukan modifikasi segi fitur dan parameter *training option*.

Modifikasi yang dilakukan berupa perubahan pada fungsi _get_features(self, tokens, idx). Perubahan yang peneliti buat berupa penampahan fitur PREV_WORD_ dan NEXT_WORD_. Berikut adalah *pseudocode* pada perubahan fungsi yang ditunjukkan pada Kode 6.1.

```
GET daftar fitur
IF idx IS NOT 0:
        tambahkan fitur (‘PREV_WORD_’ + tokens[ idx  - 1 ])
IF idx < indeks terakhir pada token:
        tambahkan fitur (‘NEXT_WORD_’ + tokens[ idx + 1 ])
```

Kode 6.1 Psudocode untuk menambahkan fitur.

Setelah melakukan penambahan fitur dilakukan uji coba model dan mendapatkan akurasi dengan perhitungan yang dapat dilihat pada Persamaan 6.2.

$$AKURASI = \frac{TRUE\ POSITIF}{TOTAL\ LABEL} = \frac{132446}{136732} = 95{,}86\%$$

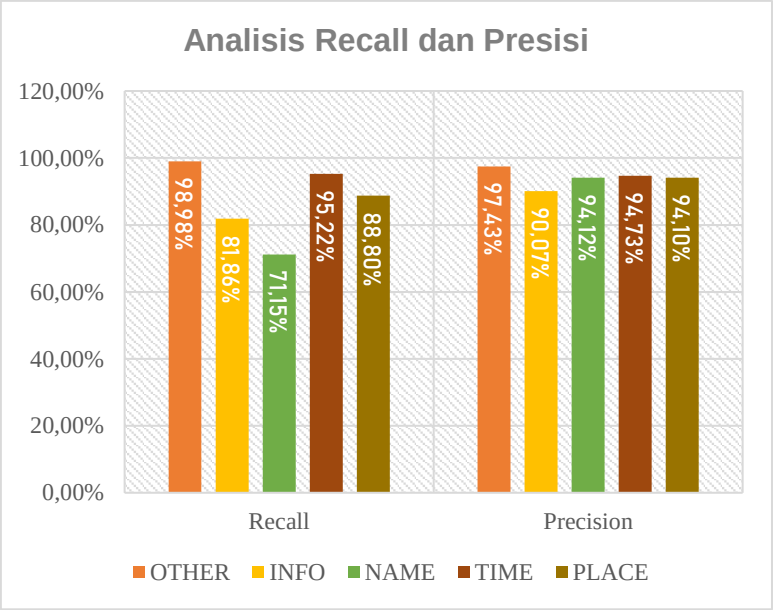
Persamaan 6.2 Perhitungan akurasi model dengan modifikasi

Dilihat dari akurasi yang baru terdapat peningkatan yang tidak terlalu signifikan. Pada sisi lain, akurasi pada model sebelumnya memang sejak awal sudah cukup tinggi. Langkah selanjutnya adalah menghitung *recall* dan presisi yang ditunjukkan pada Tabel 6.6.

Tabel 6.6 Rekap recall dan presisi per label setelah modifikasi fitur

Label	Recall	Precision
OTHER	98,98%	97,43%
INFO	81,86%	90,07%
NAME	71,15%	94,12%
TIME	95,22%	94,73%
PLACE	88,80%	94,10%
Rata-rata	**87,20%**	**94,09%**

Berikut ini adalah visualisasi dari Tabel 6.6 dapat dilihat pada Gambar 6.4.



Gambar 6.4 Diagram analisis recall dan presisi setelah modifikasi fitur

Jika dibandingkan Gambar 6.3 dengan Gambar 6.4, maka didapatkan peningkatan yang signifikan pada beberapa aspek. Terutama yang paling berdampak besar terlihat pada *recall* yang berubah secara keseluruhan sehingga mendapatkan nilai yang tidak kurang dari 70% untuk semua label. Sama halnya dengan presisi terjadi penambahan presentase pula sehingga nilai tidak ada yang kurang dari 90%. Dari *recall* dan presisi yang didapatkan maka rata-rata *recall* dan presisi adalah 87,20% dan 94,09%.

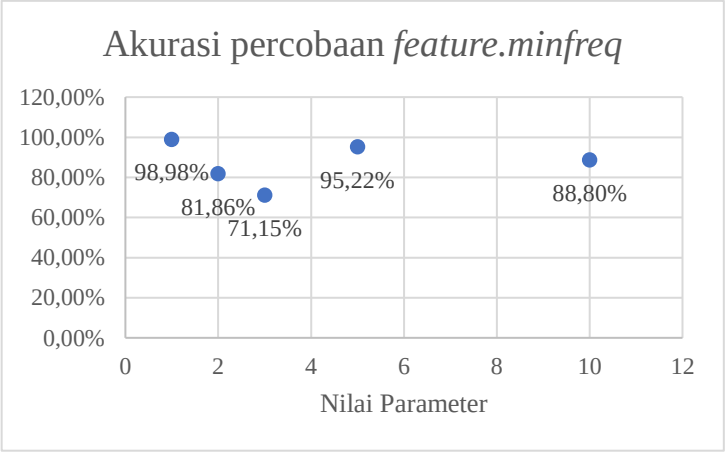
6.3.3. Uji coba model dengan parameter

Setelah mendapatkan model yang memuaskan kemudian melakukan uji coba dengan mengubah *training option* dari NLTK. Pembagian data dilakukan untuk uji coba yang biasa disebut dengan *cross validation*. Perbandingan antara jumlah data *training* dan jumlah data *testing* adalah 8:2.

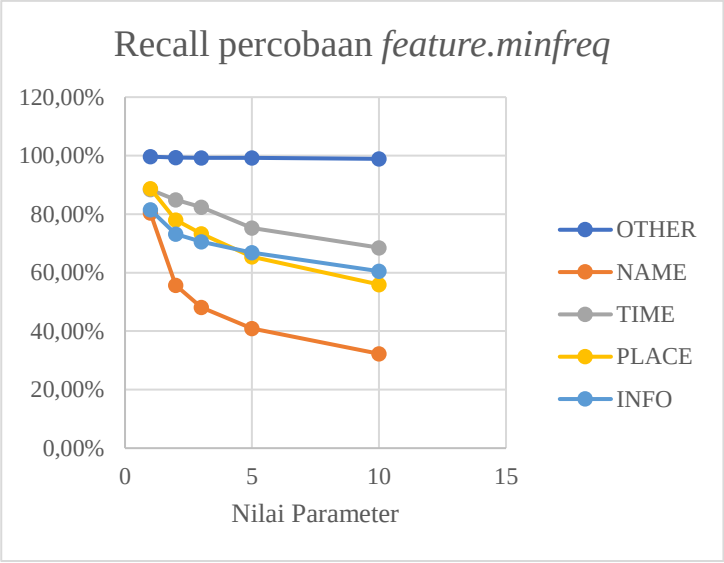
Berikut adalah hasil percobaan yang dilakukan.

a. Parameter feature.minfreq

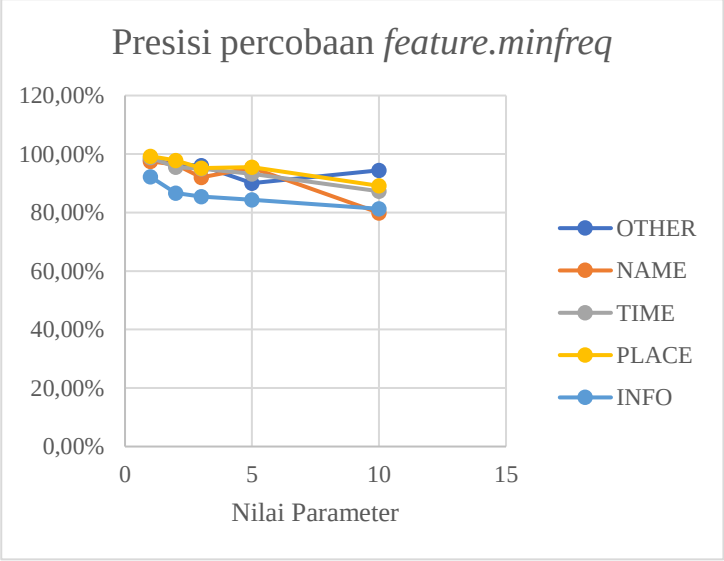
Pada parameter ini akan dilakukan uji coba menggunakan lima nilai yang berbeda antara lain 1, 2, 3, 5, dan 10. Berikut adalah perbandingan akurasi, *recall* dan presisi untuk setiap percobaan pada Gambar 6.5, Gambar 6.6, dan Gambar 6.7.



Gambar 6.5 Akurasi percobaan feature.minfreq



Gambar 6.6 Recall percobaan feature.minfreq

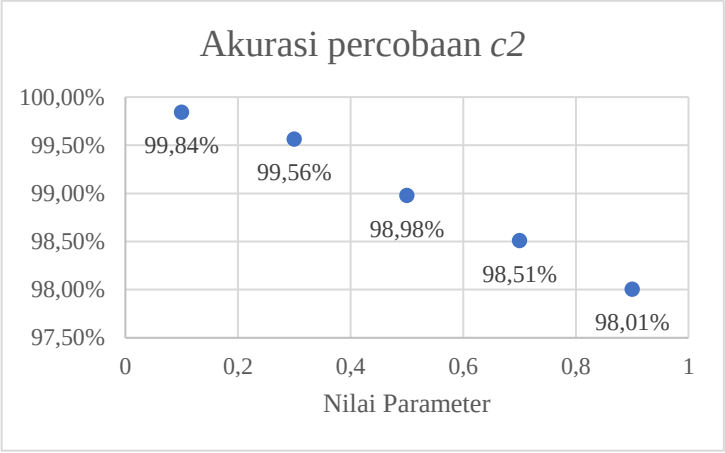


Gambar 6.7 Presisi percobaan feature.minfreq

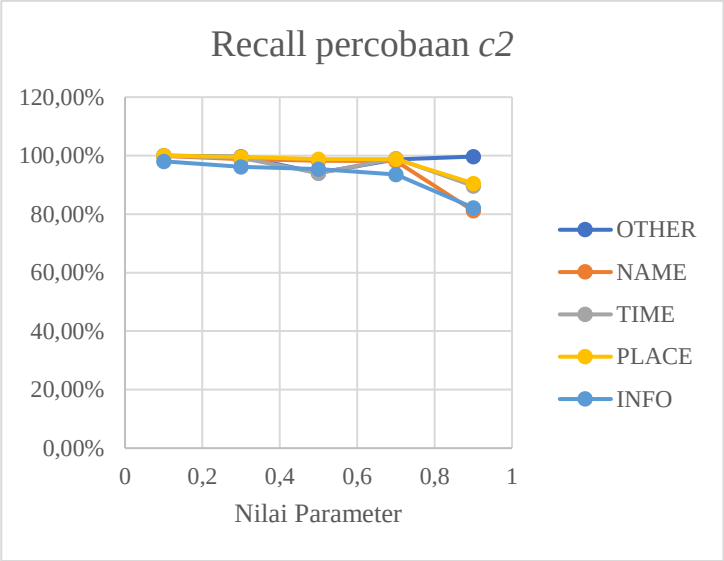
Perubahan nilai pada feature.minfreq memberikan nilai akurasi yang fluktuatuif dengan bentuk kurva pelana, sedangkan dilihat dari recall cenderung terjadi penurunan, namun untuk presisi terlihat tidak ada perubahan signifikan dan tidak memiliki kesamaan untuk semua label.

b. Parameter c2

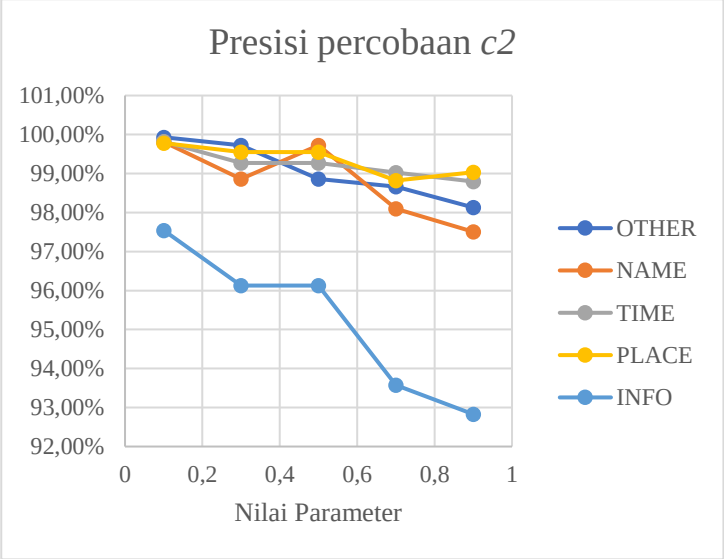
Pada parameter ini akan dilakukan uji coba menggunakan lima nilai yang berbeda antara lain 0,1; 0,3; 0,5; 0,7; dan 0,9. Berikut adalah perbandingan akurasi, *recall*, dan presisi untuk setiap percobaan pada Gambar 6.8, Gambar 6.9, dan Gambar 6.10.



Gambar 6.8 Akurasi percobaan c2



Gambar 6.9 Recall percobaan c2

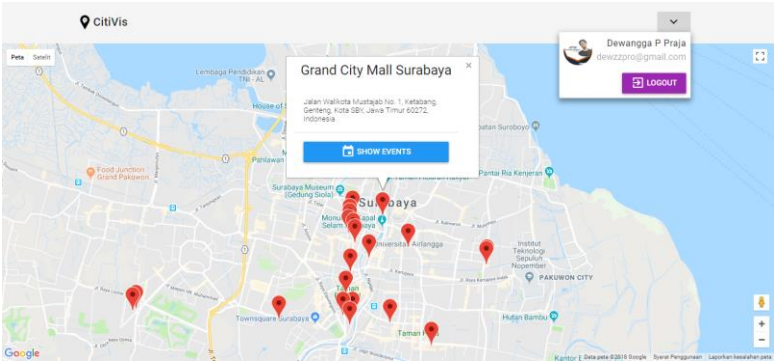


Gambar 6.10 Presisi percobaan c2

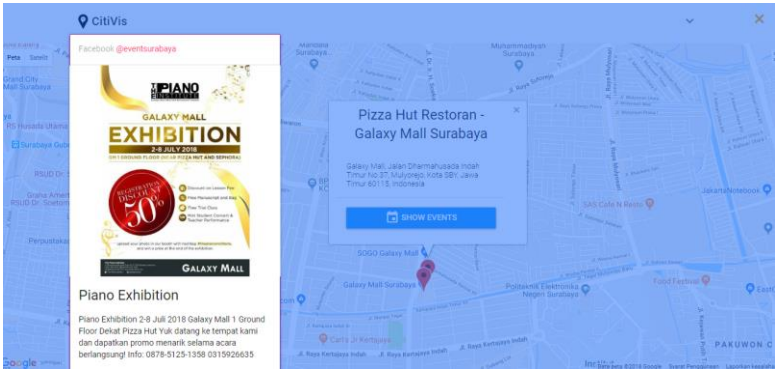
Perubahan nilai yang terjadi pada akurasi tidak berubah banyak namun ada pola menurun dari nilai 99% hingga 98% setiap kenaikan input. Untuk recall dapat dilihat terdapat penurunan dibagian akhir terutama untuk beberapa label seperti NAME dan INFO. Sedangkan untuk presisi nilai masih berkisar diatas 90% dengan pola menurun. Nilai presisi pada label INFO terlihat berbeda dibanding dengan label lain karena kurva berada dibawah label-label lain.

6.4. Hasil Tampilan Aplikasi

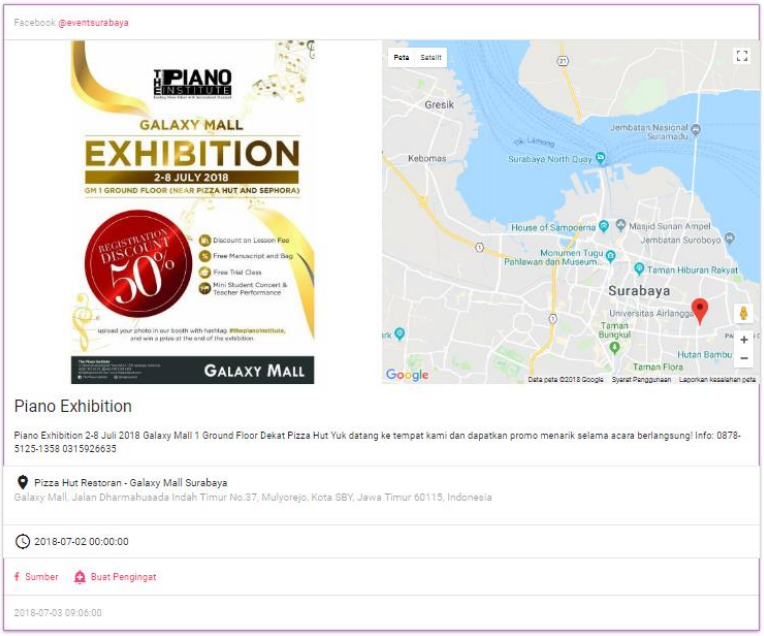
Hasil akhir dari penelitian ini adalah berupa aplikasi berbasis web yang dapat dilihat di Gambar 6.11 hingga Gambar 6.13.



Gambar 6.11 Tampilan halaman utama Aplikasi Extract Event



Gambar 6.12 Tampilan daftar event



Gambar 6.13 Tampilan detail event

Metode dalam proses pengambilan daftar event untuk suatu lokasi tertentu dilakukan menggunakan *Asynchronous JavaScript and XMLHTTP* (AJAX). Dengan menggunakan

AJAX, peta yang telah ditampilkan tidak akan berubah sehingga pengguna dapat berpindah dari halaman utama ke halaman daftar dengan cepat.

Akun Google dihubungkan langsung ke aplikasi untuk mengaktifkan fitur Buat Pengingat. Login pada sistem dilakukan dengan menekan tombol login saja dan akan redirect ke halaman yang terakhir dikunjungi. Selain itu fitur Buat Pengingat juga memberikan kemudahan pada pengguna dalam menambahkan pengingat ke akun Google secara langsung dengan praktis.

6.5. Hasil Uji Coba Aplikasi

Uji coba dilakukan dengan menjalankan 34 test dengan waktu 110,7 detik. Berikut adalah perincian dari hasil uji coba aplikasi. Hasil uji coba dapat dilihat pada LAMPIRAN B.

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini dibahas mengenai kesimpulan dari semua proses yang telah dilakukan dan saran yang dapat diberikan untuk pengembangan yang lebih baik.

7.1. Kesimpulan

Kesimpulan yang didapatkan dari proses pengerjaan tugas akhir yang telah dilakukan antara lain:

1. Pengambilan data menggunakan Apache Kafka dapat meringankan kinerja dan mengefisienkan pemrosesan data karena dikerjakan secara paralel. Hal ini dapat dilakukan berkat pemanfaatan proses threading saat pengambilan data dan pengolahan data secara bersamaan.
2. Proses pelabelan yang konsisten sangat penting untuk mendapatkan model dengan akurasi yang tinggi. Pelabelan yang dilakukan orang yang berbeda cenderung memiliki perbedaan persepsi antar para anotator dari pemahaman definisi suatu label yang menentukan bahwa token dianggap sebagai label atau tidak. Selain itu batas awal label maupun batas akhir label saat melakukan pelabelan menentukan bagaimana model merepresentasikan label.
3. *Library* NLTK yang didukung menggunakan modul CRFSuite untuk bahasa pemrograman Python belum tersedia melakukan uji coba terhadap model, sehingga menyebabkan diperlukan pembuatan mekanisme uji coba secara manual.
4. Nilai akurasi sangat tinggi hingga mencapai 94%, namun nilai tersebut tidak sepenuhnya dinilai baik. Nilai tersebut sangat dipengaruhi oleh label OTHER yang memiliki jumlah hingga dua puluh kali lipat dibanding label lain. Jika label OTHER diabaikan maka nilai akurasi yang didapatkan hanya 65%.

5. Penambahan fitur NEXT_WORD dan PREV_WORD pada modul NLTK dapat meningkatkan performa model secara signifikan ditunjukkan pada nilai *recall* dan presisi dari semua label yang meningkat pula, khususnya untuk label INFO dan NAME yang awalnya memiliki nilai 60,7% dan 49,9% menjadi 81% dan 71,15%.
6. Perubahan nilai yang terjadi saat dilakukan pengujian parameter tidak menunjukkan perbedaan yang terlalu signifikan. Untuk kedua parameter feature.minfreq dan c2, memberikan performa yang relatif menurun seiring nilai parameter yang semakin besar pula.
7. Jumlah data dan variasi bentuk data sangat mempengaruhi nilai *recall* dari label untuk setiap label. Dengan banyaknya variasi yang ditemukan maka model dapat memprediksi banyak pola susunan frase untuk suatu label.
8. Pembuatan kode yang *maintainable* sangat diperlukan dalam membuat aplikasi dengan komponen yang saling terintegrasi. Kode yang *maintainable* membantu memudahkan dalam melakukan pengubahan saat hal yang tidak diinginkan terjadi, salah satu contohnya adalah saat terjadi perubahan pada API Instagram yang menyebabkan program tidak berjalan dengan baik.

7.2. Saran

Dari pengerjaan tugas akhir ini, adapun beberapa saran untuk pengembangan penelitian ke depan.

1. Dalam melakukan pemilihan sumber data pastikan bahwa kiriman user terkait memiliki kiriman yang terkait dengan kegiatan.
2. Dibutuhkan komponen yang dapat membedakan antara kiriman yang relevan dengan membedakan antara kiriman *event* atau bukan *event*. Cara membedakan event dan bukan event adalah dengan melihat keberadaaan waktu dan tempat pada kiriman tersebut.

3. Model dapat dikembangkan lebih lanjut dengan menambahkan fitur-fitur dari token sebelumnya dan token setelahnya untuk meningkatkan dan mengetahui pengaruh fitur token sebelum dan sesudah terhadap model.
4. Aplikasi dapat dikembangkan untuk skala yang lebih luas, dapat berupa skala nasional atau berupa penambahan sumber halaman media sosial.
5. Penambahan fitur dapat dilakukan untuk meningkatkan *user experience* salah satunya adalah menambahkan fitur untuk mencari event terdekat berdasarkan lokasi pengguna.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Asosiasi Penyelenggara Jasa Internet Indonesia - APJII, "Penetrasi & Perilaku Pengguna Internet Indonesia - Survey 2016," hal. 34, 2016.

[2] H. F. A. Kumala, "Pendeteksian Nama Lokasi Dari Informasi Publik Pada Media Sosial Kota Surabaya Berbasis Named-Entity Recognition," Institut Teknologi Sepuluh Nopember, 2017.

[3] F. Muhammad dan M. L. Khodra, "Event information extraction from Indonesian tweets using conditional random field," *ICAICTA 2015 - 2015 Int. Conf. Adv. Informatics Concepts, Theory Appl.*, hal. 0–5, 2015.

[4] C. Manning, "Information Extraction and Named Entity Recognition," hal. 73, 2016.

[5] S. Liao, N. York, dan R. Grishman, "Using Document Level Cross-Event Inference to Improve Event Extraction," *Comput. Linguist.*, no. July, hal. 789–797, 2010.

[6] M. L. Khodra, "Event extraction on Indonesian news article using multiclass categorization," *ICAICTA 2015 - 2015 Int. Conf. Adv. Informatics Concepts, Theory Appl.*, hal. 1–5, 2015.

[7] J. Brownlee, "What Is Natural Language Processing?," *Machine Learning Mastery*, 2017. [Daring]. Tersedia pada: https://machinelearningmastery.com/natural-language-processing/. [Diakses: 26-Feb-2018].

[8] D. M. Neves, "Introduction to Natural Language Processing," hal. 1–25, 2016.

[9] D. Jurafsky dan J. Martin, "Information extraction," *Commun. ACM*, no. November, 2005.

[10] J. Lafferty, A. McCallum, dan F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *ICML '01 Proc. Eighteenth Int. Conf. Mach. Learn.*, vol. 8, no. June, hal. 282–289, 2001.

[11] C. Sutton dan A. McCallum, “An Introduction to Conditional Random Fields for Relational Learning,” in *Introduction to statistical relational learning*, no. x, 2006, hal. 93.

[12] E. Chen, “Introduction to Conditional Random Fields,” *http://blog.echen.me*, 2012. [Daring]. Tersedia pada: http://blog.echen.me/2012/01/03/introduction-to-conditional-random-fields/. [Diakses: 28-Feb-2018].

[13] M. Huenerfauth, G. Van Rossum, dan R. P. Muller, “Introduction to Python,” *Harvard*, 2009.

[14] Python Software Foundation, “Python 3.6.4 documentation,” 2018. [Daring]. Tersedia pada: https://docs.python.org/3/. [Diakses: 28-Feb-2018].

[15] N. Okazaki, “CRFsuite: a fast implementation of Conditional Random Fields (CRFs),” *http://www.chokkan.org*, 2007. [Daring]. Tersedia pada: http://www.chokkan.org/software/crfsuite/. [Diakses: 02-Jul-2018].

[16] N. Okazaki, “CRFsuite - CRF Benchmark test,” *http://www.chokkan.org*, 2011. [Daring]. Tersedia pada: http://www.chokkan.org/software/crfsuite/benchmark.html. [Diakses: 02-Jul-2018].

[17] S. Bird, E. Klein, dan E. Loper, *Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit*. 2014.

[18] M. S. Apostolopoulou, D. G. Sotiropoulos, I. E. Livieris, dan P. Pintelas, “A memoryless BFGS neural network training algorithm,” *7th IEEE Int. Conf. Ind. Informatics*, vol. 1, no. 2, hal. 216–221, 2009.

[19] J. Connect, “This fact sheet covers : What is Facebook ? How does it work ? Are there rules for using Facebook ? Can we control what goes on our page ?,” no. October 2013, hal. 1–7, 2014.

[20] Facebook, “Using the Graph API,” *Facebook*, 2018. [Daring]. Tersedia pada: https://developers.facebook.com/docs/graph-api/using-graph-api/v1.0. [Diakses: 28-Feb-2018].

[21] B. J. Herman, “The Ultimate Beginner ’ s Guide To Instagram,” hal. 35, 2014.

[22] Statista, “• Instagram: active users 2017 | Statista,” *Statista*, 2017. [Daring]. Tersedia pada: https://www.statista.com/statistics/253577/number-of-monthly-active-instagram-users/. [Diakses: 28-Feb-2018].

[23] V. Qazvinian, E. Rosengren, D. R. Radev, dan Q. Mei, “Rumor has it : Identifying Misinformation in Microblogs,” *Conf. Empir. Methods Nat. Lang. Process.*, hal. 1589–1599, 2011.

[24] MongoDB Incorporation, “MongoDB Documentation,” 2018. [Daring]. Tersedia pada: https://docs.mongodb.com/. [Diakses: 28-Feb-2018].

[25] Apache Software Foundation, “Introduction: Apache Kafka® is a distributed streaming platform. What exactly does that mean?,” 2017. [Daring]. Tersedia pada: https://kafka.apache.org/intro. [Diakses: 28-Feb-2018].

[26] A. Ronacher, “Flask (A Python Microframework),” *http://flask.pocoo.org/*, 2010. [Daring]. Tersedia pada: http://flask.pocoo.org/. [Diakses: 14-Mei-2018].

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Penulis lahir di Kabupaten Semarang pada tanggal 15 November 1995. Merupakan anak kedua dari dua bersaudara. Penulis telah menempuh beberapa pendidikan formal yaitu; SD Negeri 1 Ungaran, SMP Negeri 1 Ungaran, dan SMA Negeri 1 Salatiga.

Pada tahun 2014 pasca kelulusan SMA, penulis melanjutkan pendidikan dengan jalur SNMPTN di Jurusan Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi – Institut Teknologi Sepuluh Nopember (ITS) Surabaya dan terdaftar sebagai mahasiswa dengan NRP 0521 14 4000 0038. Selama menjadi mahasiswa, penulis mengikuti berbagai kegiatan kemahasiswaan seperti beberapa kepanitiaan serta pernah menjabat sebagai Staf Departemen Aplikasi Teknologi HMSI FTIK ITS dan Staf Kementerian Sosial Masyarakat BEM ITS. Selain itu, pada tahun ketiga mengikuti berbagai kegiatan kemahasiswaan seperti beberapa kepanitiaan dan pernah menjabat menjadi Sekretaris Departemen Riset dan Teknologi HMSI FTIK ITS. Sedangkan, pada tahun keempat pernah menjabat menjadi Koordinator Wilayah Surabaya Komunitas Sobat Bumi Indonesia. Di bidang akademik, penulis aktif menjadi asisten dosen pada beberapa mata kuliah seperti Bahasa Pemrograman, Algoritma dan Struktur Data, Pemrograman Berbasis Web, dan Sistem Cerdas.

Pada tahun keempat, penulis memiliki ketertarikan di bidang pengolahan serta visualisasi data maka dari itu penulis mengambil bidang minat Akuisisi Data dan Diseminasi Informasi (ADDI). Penulis dapat dihubungi melalui email di dewanggappraja@gmail.com.

Halaman ini sengaja dikosongkan

LAMPIRAN A

Dokumentasi Class MongoDB

__init__(self, config : str, config_name : str, db_name : str, dataset : str)	
Menginisialisasi kelas MongoDB yang mengimplementasikan AbstractDB	
Parameter	
<str> config Nama file configurasi. <str> config_name Nama configuration identifier dalam file config. <str> db_name Nama database pada host MongoBD. Nilai default adalah variable dari DEFAULT_DB_NAME <str> dataset Nama dataset dimana data akan disimpan. Dapat diubah menggunakan setDataset.	
_check_status(self) : void	
Mengecek status dari MongoDB Client. Jika pengaturan tidak ditemukan maka akan mencetak “no mongo” pada console.	
Parameter	**Return**
setDataset(self, dataset : str) : void	
Mengubah dataset tujuan.	
Parameter	**Return**
<str> dataset String nama dataset.	
putData(self, data : dict) : void	
Menginisialisasi kelas MongoDB yang mengimplementasikan AbstractDB	
Parameter	**Return**

<dict> data Data yang ingin disimpan. Minimal berisi key ID. Contoh: { '_id':'123456789' }	

getEntries(self, offset : int, limit : int) : cursor

Mengambil serangkauan data menggunakan batas awal dan batas jumlah

Parameter	Return
<int> offset Indeks mulainya rangkaian entri. <int> limit Jumlah data entri yang ingin diambil.	<pymongo.cursor.Cursor> Serangkaian data dari database.

getAll(self) : cursor

Mengambil semua data pada dataset.

Parameter	Return
	<pymongo.cursor.Cursor> Serangkaian data dari database.

getId(self, id : str) : cursor

Mengambil semua data pada dataset.

Parameter	Return
<str> id String data ID.	<pymongo.cursor.Cursor> Satu data dari database.

getTimestamp(self, id : str) : datetime

Mengambil timestamp dari satu data.

Parameter	Return
<str> id String data ID.	<datetime.datetime> Timestamp dari data yang dipilih.

setTimestamp(self, id : str) : void

Mengubah timestamp dari satu data.	
Parameter	**Return**
<str> id String data ID.	

setData(self, data : dict) : void	
Mengubah timestamp dari satu data.	
Parameter	**Return**
<dict> data Data berisi ID, indeks label, dan nama label Contoh: {'_id':'123456789', 'index':9, 'tag':'B-NAME' }	

setType(self, id : str, type : str) : void	
Mengubah kategori dari satu data.	
Parameter	**Return**
<str> id String data ID. <str> type Nama kategori.	

removeType(self, id : str) : void	
Menghapus kategori dari satu data.	
Parameter	**Return**
<str> id String data ID.	

setDataset(self, dataset : str) : void	
Mengubah nama dataset.	
Parameter	**Return**
<str> dataset String nama dataset.	

getDataset(self) : MongoClient	
Menghapus kategori dari satu data.	
Parameter	**Return**

	<MongoClient> MongoClient yang sudah diinisiasi nama database dan datasetnya.

Dokumentasi Class Instagram

__init__(self, configFile : str, limit : int, pageIds : list)	
Menginisialisasi kelas Instagram	
Parameter	
<str> configFile Nama file configurasi. Nilai default adalah ‘config.ini’ <int> limit Batas iterasi untuk setiap ID halaman. Setiap sekali iterasi terdiri dari 20 data. Nilai default adalah 1000 iterasi. <list> pageIds Daftar ID halaman sebagai target scrapping. Nilai default adalah ['acarasurabaya', 'eventpemudasurabaya', 'eventsurabaya', 'info_surabaya'].	
connect(self) : void	
Menghubungkan ke host MySQL berdasarkan file configuration yang telah diinisialisasi sebelumnya. Perlu dipanggil sekali saat crawler proses berjalan.	
Parameter	**Return**
disconnect(self) : void	
Memutuskan hubungan dari host MySQL. Perlu dipanggil sekali setelah proses crawler selesai.	
Parameter	**Return**
countData(self) : int	
Mengambil jumlah data yang yang telah di crawling dari proses yang berjalan.	
Parameter	**Return**
	<int>

	Panjang daftar data yang telah dicrawling

getRequest(self, request : str, loop : int, page_id : str) : void

Mengirimkan satu permintaan ke API Instagram untuk mendapatkan balasan JSON data.

Parameter	Return
<str> request Link URL yang akan dituju <int> loop Urutan iterasi dari request yang telah dibuat. <str> page_id String ID kiriman.	

insert(self, current_post : object, page_id : str) : void

Menambahkan satu kiriman data ke MySQL.

Parameter	Return
<dict> current_post Satu data dari kiriman yang berasal dari json data. Data berisi ID halaman, caption, kode link Instagram, URL tampilan, dan tanggal pengiriman. <str> page_id String ID kiriman.	

start(self) : void

Fungsi utama yang perlu dipanggil dari kelas Instagram ini.

```
>>> import ig
>>> instagram = ig.Instagram()
>>> instagram.start()
```

Parameter	Return

Dokumentasi Class Facebook

__init__(self, configFile : str, limit : int, pageIds : list)	
Menginisialisasi kelas Facebook	
Parameter	
<str> configFile Nama file configurasi. Nilai default adalah ‘config.ini’ <int> limit Batas iterasi untuk setiap ID halaman. Setiap sekali iterasi terdiri dari 20 data. Nilai default adalah 1000 iterasi. <list> pageIds Daftar ID halaman sebagai target scrapping. Nilai default adalah ['eventsurabaya', 'gmoevent2015'].	
connect(self) : void	
Menghubungkan ke host MySQL berdasarkan file configuration yang telah diinisialisasi sebelumnya. Perlu dipanggil sekali saat crawler proses berjalan.	
Parameter	**Return**
disconnect(self) : void	
Memutuskan hubungan dari host MySQL. Perlu dipanggil sekali setelah proses crawler selesai.	
Parameter	**Return**
getRequest(self, request : str, page_id : str) : object	
Mengirimkan satu permintaan ke API Facebook.	
Parameter	**Return**
<str> request Link URL yang akan dituju. <str> page_id String ID kiriman.	<dict> Balasan JSON data. Digunakan untuk mengambil link URL halaman berikut.
insert(self, current_post : object, page_id : str) : void	
Menambahkan satu kiriman data ke MySQL.	

Parameter	Return
<dict> current_post Satu data dari kiriman yang berasal dari json data. Data berisi ID halaman, pesan, tautan, dan waktu dibuat. <str> page_id String ID kiriman.	

start(self) : void

Fungsi utama yang perlu dipanggil dari kelas Facebook ini.

```
>>> import fb
>>> pageIds = ['eventsurabaya', 'gmoevent2015']
>>> facebook = fb.Facebook(pageIds = pageIds)
>>> facebook.start()
```

Parameter	Return

Dokumentasi Class Tagging

__init__(self, configFile : str, table_name : str)

Menginisialisasi kelas Tagging

Parameter
<str> configFile Nama file configurasi. <str> table_name Nama tabel pada database MySQL yang menyimpan data hasil crawling. Nilai default ‘posts’

connect(self) : void

Menghubungkan ke host MySQL berdasarkan file configuration yang telah diinisialisasi sebelumnya. Perlu dipanggil sekali saat proses berjalan.

Parameter	Return

disconnect(self) : void

Memutuskan hubungan dari host MySQL. Perlu dipanggil sekali setelah proses selesai.	
Parameter	**Return**

countdata(self) : list

Mengambil jumlah data dari tabel data crawler yang telah diinisiasi. Di kelompokan berdasarkan sumber dan user ID.	
Parameter	**Return**
	<list> daftar dari hasil perhitungan jumlah berdasarkan sumber dan user ID.

countdataset(self, version : int) : list

Versi dari countdata(self) namun perlu di versi terbaru dari dataset untuk mengambil jumlah data yang telah dikelompokkan berdasar sumber dan user ID.	
Parameter	**Return**
<str> version Angka versi terbaru	<list> daftar dari hasil perhitungan jumlah berdasarkan sumber dan user ID.

createdataset(self, limit : int) : void

Membuat tabel dengan nama ‘datasets’ yang berisi ID dan versi. Versi diinisiasi dengan nilai 0.	
Parameter	**Return**
<str> limit Jumlah data yang diinisiasi untuk pertama kali dari setiap sumber dengan user yang berbeda	

createtag(self, name : str) : void

Menambahkan data yang memiliki ID tercantum pada tabel ‘datasets’ dari database MySQL ke MongoDB. createdataset(self) harus pernah dijalankan terlebih dahulu.	
Parameter	**Return**
<str> name Nama dataset pada MongoDB	

tokenize(self, data : dict, dataset_name : str, reset : boolean) : void	
Melakukan tokenisasi kemudian memasukkan data dari tabel sumber ke MongoDB.	
Parameter	**Return**
<dict> data Data yang akan ditokenisasi dan ditambahkan. <str> dataset_name Nama dataset di MongoDB. <boolean> reset Jika reset memiliki nilai True maka proses akan mengubah semua label menjadi 'O' jika sebelumnya sudah ada. Jika bernilai False maka proses akan mengabaikan data sebelumnya yang sudah ada. Nilai default adalah False.	

newdataset(self, limit : int, version : int) : void	
Menambahkan ID dari data terbaru ke tabel sumber dengan nama 'datasets' yang berisi ID dan versi.	
Parameter	**Return**
<int> limit Jumlah data yang diinisiasi untuk untuk setiap sumber dengan user yang berbeda <int> version Angka versi terbaru	

Dokumentasi Class DataAdapter

__init__(self, data : list/pymongo.cursor.Cursor)

Menginisialisasi kelas DataAdapter	
Parameter	
<list> <pymongo.cursor.Cursor> data Daftar data yang ingin diproses. Lebih baik menginisialisasikan banyak data pada fungsi ini. Untuk data yang tidak banyak diinisialisasi pada fungsi set_data(self).	
count(self, data : list/pymongo.cursor.Cursor)	
Menghitung data jika data bertipe list atau pymongo.cursor.Cursor.	
Parameter	**Return**
<list> <pymongo.cursor.Cursor> data Daftar data yang ingin diproses.	<int> Jumlah data.
set_data(self, data : list) : void	
Menginisiasi data baru dan akan menimpa data yang sebelumnya terinisialisasi.	
Parameter	**Return**
<list> <pymongo.cursor.Cursor> data Daftar data yang ingin diproses.	
tokenize_tag(self, text : str) : list, list	
Melakukan tokenisasi pada pesan kemudian melakukan memprediksikan label untuk setiap token dengan menggunakan model.	
Parameter	**Return**
<str> text Pesan teks yang ingin diproses.	<list> Daftar token yang ditokenisasi <list> Daftar label yang diprediksi dengan model
for_tagging(self, data : list) : list	
Mengubah format data menjadi format yang dapat diproses dan siap untuk dilabel.	

Parameter	Return
<list> <pymongo.cursor.Cursor> data Daftar data yang ingin diproses.	<list> Daftar token yang ditokenisasi.

for_testing(self, data : list) : list

Mengubah format data menjadi format yang dapat diproses dan siap untuk dilabel.

Parameter	Return
<list> <pymongo.cursor.Cursor> data Daftar data yang ingin diproses.	<list> Daftar token dan label.

for_tagging_testing(self, , data : list) : list, list

Menggabungkan dua fungsi sekaligus namun return value disimpan untuk digunakan pada tahap selanjutnya, seperti training dan testing.

Parameter	Return
<list> <pymongo.cursor.Cursor> data Daftar data yang ingin diproses.	<list> Daftar token yang ditokenisasi. <list> Daftar token dan label.

tag(self, data : list) : list

Proses labeling satu data menggunakan model yang telah dibuat sebelumnya.

Parameter	Return
<list> data Daftar data yang ingin diproses. Format data berupa hasil kembalian yang ada pada fungsu for_tagging(self, data)	<list> Daftar label yang dilabel menggunakan model.

labeling(self) : list

Proses labeling daftar data yang diinisialisasi pada proses sebelumnya.	
Parameter	**Return**
	<list> Daftar label yang dilabel menggunakan model.

evaluating(self) : void
Mengevaluasi model berdasarkan data testing yang telah diproses sebelumnya. Hasil akurasi akan tercetak pada console.

Parameter	**Return**

tag(self, data : list) : list
Proses labeling satu data menggunakan model yang telah dibuat sebelumnya.

Parameter	**Return**
<list> data Daftar data yang ingin diproses. Format data berupa hasil kembalian yang ada pada fungsu for_tagging(self, data). Jika tidak ada data sebagai parameter yang pada fungsi ini maka akan memproses data yang sebelumnya diinisialisasi. Nilai default adalah None.	

Dokumentasi Class Crawl

__init__(self, client : KafkaClient, topic : str, name : str, producer : Producer, delay : int)
Menginisialisasi kelas Crawl yang diturunkan dari kelas threading.Thread.
Parameter
<KafkaClient> client

Menginisiasi klien kafka yang digunakan untuk membuat producer.

<str> topic
Topic klien sebagai tujuan kemana pesan perlu dikirim.

<Producer> producer
Menginisiasikan produser apabila diperlukan.

<int> delay
Berapa lama perlu menunggu dari satu permintaan ke permintaan berikutnya ke server.

<str> name
Nama thread untuk proses yang dijalankan.

get_producer(self, topic : str, sync : boolean, use_rdkafka : boolean, delivery_reports : boolean, max_request_size : int, **kwargs) : Producer

Mengambil produser atau menginisialisasi produser menggunakan klien apabila belum diinisialisasi.

Parameter	Return
<str> topic Topic klien sebagai tujuan kemana pesan perlu dikirim. <boolean> sync Salah satu parameter yang diinisialisasi pada produser oleh kafka. Nilai default adalah True. <boolean> use_rdkafka Salah satu parameter yang diinisialisasi pada produser oleh kafka. Nilai default adalah False. <boolean> delivery_reports Salah satu parameter yang diinisialisasi pada produser	<Producer> producer Producer yang telah diinisialisasi.

oleh kafka. Nilai default adalah True. <int> max_request_size Salah satu parameter yang diinisialisasi pada produser oleh kafka. Nilai default adalah 5MB.	

stop(self) : void	
Permintaan untuk menghentikan thread.	
Parameter	**Return**

run(self) : void	
Menjalankan fungsi ini jika thread mulai berjalan. Default behaviournya adalah menjalankan secara realtime, overide fungsi ini untuk mengimplementasikan behaviour lain.	
Parameter	**Return**

run_onetime(self) : void	
Fungsi ini perlu dioviride untuk mengimplementasikan behaviournya.	
Parameter	**Return**

run_realtime(self) : void	
Bentuk looping untuk fungsi run onetime.	
Parameter	**Return**

producing(self, value : dict, key : str) : void	
Mengirimkan pesan oleh Kafka Producer	
Parameter	**Return**
<dict> value Data yang perlu dikirim. <str> key Primary key untuk pesan yang akan dikirim.	

serializer(self) : bytes, bytes	
Kafka hanya bisa mengirim dalam bentuk bytes, sehingga pesan perlu diubah menjadi bytes menggunakan fungsi ini,	
Parameter	**Return**

<dict> value Data yang perlu dikirim. <str> key Primary key untuk pesan yang akan dikirim.	<bytes> Data dalam bentuk bytes. <bytes> Primary key dalam bentuk bytes.

connect(self) : void	
Menghubungkan ke host MySQL berdasarkan file configuration yang telah diinisialisasi sebelumnya. Perlu dipanggil sekali saat proses berjalan.	
Parameter	**Return**

disconnect(self) : void	
Memutuskan hubungan dari host MySQL. Perlu dipanggil sekali setelah proses selesai.	
Parameter	**Return**

getNewestID(self, source : str, page_id : str) : int	
Mengambil ID terbaru dari sumber dan dan halaman spesifik.	
Parameter	**Return**
<str> source Nama media sosial. ‘tw’ untuk twitter, ‘fb’ untuk facebook, ‘ig’ untuk instagram <str> page_id String ID halaman.	<int> ID data terbaru berupa integer dari database.

remove(self, id : str) : void	
Menghapus satu data dengan ID spesifik.	
Parameter	**Return**
<str> id String data ID.	

Dokumentasi Class CrawlFacebook

__init__(self, client : KafkaClient, account : str, topic : str, **kwargs)

Menginisialisasi kelas CrawlFacebook yang mengimplementasikan kelas Crawl.
Parameter
<KafkaClient> client Menginisiasi klien kafka yang digunakan untuk membuat producer. <str> topic Topic klien sebagai tujuan kemana pesan perlu dikirim. <str> account ID halaman facebook yang akan dicrawling.

run(self) : void	
Menjalankan fungsi ini jika thread mulai berjalan. Default behaviournya adalah menjalankan fungsi run(self) yang ada di kelas Crawl.	
Parameter	**Return**

run_onetime(self) : void	
Mengecek update sekali dari halaman Facebook.	
Parameter	**Return**

getRequest(self, request : str, page_id : str, since_id : int) : str, str	
Mengirimkan satu permintaan ke Facebook API untuk halaman tertentu dengan setelah post ID tertentu.	
Parameter	**Return**
<str> request Link url yang akan dikirim ke API. <str> page_id ID halaman Facebook. <int> since_id ID kiriman untuk membatasi jumlah post dalam satu permintaan. ID kiriman adalah kiriman terbaru yang sudah ada pada database.	<str> Max ID untuk membatasi kiriman. <str> ID batas awal data yang diambil untuk satu request.

<int> max_id ID kiriman untuk membatasi jumlah post dalam satu permintaan. ID kiriman yang digunakan untuk mengiterasi antara batas kiriman setiap satu permintaan.	

getNewestID(self, page_id : str) : str	
Mengambil ID kiriman terbaru yang tersimpan pada database.	
Parameter	**Return**
<str> page_id ID halaman Facebook.	<str> ID terakhir yang tersimpan di database.

insert(self, current_post : dict, page_id : str) : void	
Mengambil ID kiriman terbaru yang tersimpan pada database.	
Parameter	**Return**
<dict> current_post Data yang ingin disimpan. <str> page_id ID halaman Facebook.	

Dokumentasi Class CrawlTwitter

__init__(self, client : KafkaClient, account : str, topic : str, **kwargs)
Menginisialisasi kelas CrawlTwitter yang mengimplementasikan kelas Crawl.
Parameter
<KafkaClient> client Menginisiasi klien kafka yang digunakan untuk membuat producer. <str> topic

Topic klien sebagai tujuan kemana pesan perlu dikirim. <str> account ID halaman twitter yang akan dicrawling.	
run(self) : void	
Menjalankan fungsi ini jika thread mulai berjalan. Default behaviournya adalah menjalankan fungsi run(self) yang ada di kelas Crawl.	
Parameter	**Return**
run_onetime(self) : void	
Mengecek update sekali dari akun Twitter.	
Parameter	**Return**
getRequest(self, page_id : str, since_id : int, max_id : int) : cursor	
Mengirimkan satu permintaan ke Twitter API untuk halaman tertentu dengan setelah post ID tertentu.	
Parameter	**Return**
<str> page_id ID halaman Twitter. <int> since_id ID kiriman untuk membatasi jumlah post dalam satu permintaan. ID kiriman adalah kiriman terbaru yang sudah ada pada database. <int> max_id ID kiriman untuk membatasi jumlah post dalam satu permintaan. ID kiriman yang digunakan untuk mengiterasi antara batas kiriman setiap satu permintaan.	<str> ID yang terakhir tersimpan di database.
getNewestID(self, page_id : str) : str	
Mengambil ID kiriman terbaru yang tersimpan pada database.	

Parameter	Return
<str> page_id ID halaman Twitter.	<str> ID terakhir yang tersimpan di database.

insert(self, current_post : object, page_id : str) : void

Mengambil ID kiriman terbaru yang tersimpan pada database.

Parameter	Return
<dict> current_post Data yang ingin disimpan. <str> page_id ID halaman Facebook.	

Dokumentasi Class DataProvider

__init__(self, config : str, mapClient : googlemaps.Client, mongo : MongoDB)

Menginisialisasi kelas DataProvider

Parameter

<str> config
Nama file configurasi. Nilai default adalah ‘config.ini’

<googlemaps.Client> mapClient
Klien yang digunakan googlemaps API. Nilai default adalah None.

<database.md.MongoDB> MongoDB
Modul mongoDB yang digunakan untuk mengambil data. Jika tidak diinisialisasi maka akan menginisiali MongoDB sendiri.

setMapClient(self, mapClient : googlemaps.Client)

Mengubah klient untuk googlemaps API.

Parameter	Return
<googlemaps.Client> mapClient	

Klien yang digunakan googlemaps API. Nilai default adalah None.	

getData(self, id : str, save : boolean) : dict

Mendapatkan satu data dengan ID tertentu dan menambahkan satu key value yang berisikan entitas dari data yang diambil.

Parameter	Return
<str> id String data ID <boolean> save Jika True maka simpan data yang diambil akan disimpan ke database. Nilai default False.	<dict> Data dari database dengan tambahan entities sebagai key value.

extractText(self, labels : list, text : list) : list

Memproses serangkaian label dan token untuk di ekstrak label pentingnya saja.

Parameter	Return
<list> labels Daftar label yang ingin diekstrak <list> text Daftar token yang telah dilabeli	<list> Daftar label penting yang sudah diekstraksi Contoh: [{'label': 'NAME', 'text': ['lorem','ipsum','dolor']}, {'label': 'PLACE', 'text': ['Jl.','ipsum','95']}]

removeDuplicate(self, old_list : list, key : str) : list

Menghapus data dengan tipe dict yang kembar dalam suatu list.

Parameter	Return
<list> old_list Daftar data dict. <str> key	<list> Daftar yang baru tanpa data dict yang kembar.

Key dari dict yang mengidentifikasikan bahwa data tersebut unik.	

parserEvent(self, label : str, array : list, created : datetime) : str/datetime

Mengubah bentuk token menjadi tipe data lain berdasarkan labelnya.

Parameter	Return
<str> label Nama label. <list> array Daftar token.	<str> <list> <datetime.datetime> Jika label adalah TIME maka akan mengembalikan nilai berupa datetime.datetime. Jika label adalah PLACE maka akan mengembalikan nilai berupa dict yang berisi informasi lokasi. Selain itu akan mengembalikan nilai berupa gabungan token sebagai satu rangkaian teks.

time_process(self, text : str, created : datetime) : datetime

Mengubah string menjadi datetime.

Parameter	Return
<str> old_list String yang akan diubah. <datetime.datetime> created_time Nilai yang menjadi patokan relatif dari string yang akan diubah.	<datetime.datetime> Tanggal hasil perubahan dari teks.

place_process(self, text : str) : list

Mencari daftar lokasi berdasarkanfitur pencarian yang disediakan Google Maps API.

Parameter	Return
<str> old_list String yang dicari.	<datetime.datetime> Daftar data informasi tempat yang didapatkan dari API.

<datetime.datetime> created_time Nilai yang menjadi patokan relatif dari string yang akan diubah.	

LAMPIRAN B

Hasil Test Case mengambil satu data dengan ID spesifik

No.	: 101
Scenario	: Mengambil data dengan menggunakan modul MongoDB
File Target	: database/md.py
Last Run	: 02/07/18 22:26
Duration	: 5,67 detik
Test Case No.	: 001
Test Name	: test.md_test.GetMongoTestCase.test_get_id
Test Case Description	: Mengambil satu data dengan ID spesifik

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	id masukkan	nilai _id pada variable result	☑
2	AssertEqual	mockup daftar label	nilai label pada variable result	☑
3	AssertEqual	mockup daftar text	nilai text pada variable result	☑
4	AssertEqual	mockup waktu dibuat dengan tipe data datetime	nilai created time pada variable result	☑
5	AssertEqual	mockup url sumber	nilai source_url pada variable result	☑
6	AssertEqual	mockup url gambar	nilai image_url pada variable result	☑

Hasil Test Case menginput data dan mengecek kembali dari database

No.	: 102
Scenario	: Memasukkan data ke database menggunakan modul MongoDB
File Target	: database/md.py
Last Run	: 03/07/18 03:53
Duration	: 9,21 detik
Test Case No.	: 001

Test Name		: test.md_test.PutMongoTestCase.test_put_data_0		
Test Case Description		: Menginput data dan mengecek kembali dari database		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	data yang telah di input ke database	data yang diambil menggunakan input id	☑
2	AssertEqual	value dengan tipe datetime dari timestamp pada data yang telah diinput	timestamp yang diambil menggunakan input id	☑

Hasil Test Case menginputkan data lebih dari sekali dengan ID yang sama

No.		: 102		
Scenario		: Memasukkan data ke database menggunakan modul MongoDB		
File Target		: database/md.py		
Last Run		: 03/07/18 22:26		
Duration		: 9,21 detik		
Test Case No.		: 002		
Test Name		: test.md_test.PutMongoTestCase.test_put_data_1		
Test Case Description		: Menginput data lebih dari sekali dengan ID yang sama		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	data pertama yang di input ke database	data yang diambil menggunakan input id	☑
2	AssertEqual	data yang diambil menggunakan input id	data yang diambil menggunakan input id	☑
3	AssertEqual	jumlah data pada database hanya 1	jumlah semua data yang diambil dari database	☑

Hasil Test Case menginput data dan mengambil data menggunakan limit dan offset

No.	: 102
Scenario	

	: Memasukkan data ke database menggunakan modul MongoDB
File Target	: database/md.py
Last Run	: 03/07/18 04:11
Duration	: 5,63 detik
Test Case No.	: 003
Test Name	: test.md_test.PutMongoTestCase.test_put_data_2
Test Case Description	: Menginput data dan mengambil data menggunakan limit dan offset

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	jumlah data yang terambil adalah 2	jumlah data yang diambil menggunakan offset 1 dan limit 2	☑
2	AssertEqual	data input pada indeks 1 sampai indeks 3	bentuk list dari hasul data yang diambil menggunakan offset 1 dan limit 2	☑

Hasil Test Case mengubah data label dengan data ID dan indeks tertentu

No.	: 103
Scenario	: Mengubah data dari database menggunakan modul MongoDB
File Target	: database/md.py
Last Run	: 03/07/18 04:16
Duration	: 5,90 detik
Test Case No.	: 001
Test Name	: test.md_test.SetMongoTestCase.test_set_data_0
Test Case Description	: Mengubah data label dengan data ID dan indeks tertentu

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	data label yang di inputkan	value label dari data yang diambil pada data indeks yang diinputkan	☑

Hasil Test Case mengubah data label dengan data ID yang sesuai namun input lain salah

No.	: 103
Scenario	: Mengubah data dari database menggunakan modul MongoDB
File Target	: database/md.py
Last Run	: 03/07/18 04:20
Duration	: 4,96 detik
Test Case No.	: 002
Test Name	: test.md_test.SetMongoTestCase.test_set_data_1
Test Case Description	: Mengubah data label dengan data ID yang sesuai namun input lain salah

Assertions

No	Assertion	Expected	Actual	Result
1	AssertRaises	KeyError	saat melakukan pengubahan data	☑

Hasil Test Case mengubah data label dengan input yang salah

No.	: 103
Scenario	: Mengubah data dari database menggunakan modul MongoDB
File Target	: database/md.py
Last Run	: 03/07/18 04:20
Duration	: 4,96 detik
Test Case No.	: 003
Test Name	: test.md_test.SetMongoTestCase.test_set_data_2
Test Case Description	: Mengubah data label dengan input yang salah

Assertions

No	Assertion	Expected	Actual	Result
1	AssertRaises	KeyError	saat melakukan pengubahan data	☑

Hasil Test Case mengubah tipe data

No.	: 103
Scenario	: Mengubah data dari database menggunakan modul MongoDB
File Target	: database/md.py
Last Run	: 03/07/18 04:24

Duration		: 5,59 detik		
Test Case No.		: 004		
Test Name		: test.md_test.SetMongoTestCase.test_set_type_0		
Test Case Description		: Mengubah data pada type		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	tipe yang diinputkan	value tipe pada data yang diambil	☑

Hasil Test Case mengubah data type kemudian menghapus kembali

No.		: 103		
Scenario		: Mengubah data dari database menggunakan modul MongoDB		
File Target		: database/md.py		
Last Run		: 03/07/18 04:20		
Duration		: 4,96 detik		
Test Case No.		: 005		
Test Name		: test.md_test.SetMongoTestCase.test_set_type_1		
Test Case Description		: Mengubah data pada type kemudian menghapus kembali		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	tipe yang diinputkan	value tipe pada data yang diambil	☑
2	AssertRaises	KeyError	saat melakukan mengambil value type dari data yang diambil	☑

Hasil Test Case mengubah data type dengan ID yang salah

No.	: 103
Scenario	: Mengubah data dari database menggunakan modul MongoDB
File Target	: database/md.py
Last Run	: 03/07/18 04:32
Duration	: 5,51 detik
Test Case No.	: 006
Test Name	: test.md_test.SetMongoTestCase.test_set_type_2
Test Case Description	

Assertions		: Mengubah data pada type dengan menggunakan ID yang salah		
No	Assertion	Expected	Actual	Result
1	AssertEqual	None	data yang diambil menggunakan id	☑

Hasil Test Case memasukkan banyak data kemudian menghapus yang memiliki message yang sama

No.		: 104		
Scenario		: Menghapus data dengan teks yang sama menggunakan modul MongoDB		
File Target		: database/md.py		
Last Run		: 03/07/18 05:04		
Duration		: 6,58 detik		
Test Case No.		: 001		
Test Name		: test.md_test.RemoveDuplicateTestCase.test_remove		
Test Case Description		: Memasukkan banyak data kemudian menghapus yang memiliki full_text yang sama		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	jumlah data yang diinputkan ke database adalah 5	jumlah semua data yang diambil dari database	☑
2	AssertEqual	jumlah data yang ada di database adalah 1 setelah dilakukan penghapusan	jumlah semua data yang diambil dari database	☑

Hasil Test Case mengubah teks menjadi waktu dengan tipe datetime, format tanggal bulan tahun

No.	: 201
Scenario	: Melakukan pengolahan pada data waktu
File Target	: data/provider.py
Last Run	: 03/07/18 09:31
Duration	: 2,41 detik
Test Case No.	: 001
Test Name	: test.provider_test.TimeTestCase.test_time_process_0
Test Case Description	

Assertions		: Mengubah teks menjadi waktu dengan tipe datetime, format tanggal bulan tahun		
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data dengan tipe datetime	Hasil pengolahan waktu dengan input teks	☑

Hasil Test Case mengubah teks menjadi waktu dengan tipe datetime, format tanggal-bulan-tahun

No.		: 201		
Scenario		: Melakukan pengolahan pada data waktu		
File Target		: data/provider.py		
Last Run		: 03/07/18 09:34		
Duration		: 1,48 detik		
Test Case No.		: 003		
Test Name		: test.provider_test.TimeTestCase.test_time_process_2		
Test Case Description		: Mengubah teks menjadi waktu dengan tipe datetime, format tanggal-bulan-tahun		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data dengan tipe datetime	Hasil pengolahan waktu dengan input teks	☑

Hasil Test Case mengubah teks menjadi waktu dengan tipe datetime, format bulan tanggal, tahun

No.		: 201		
Scenario		: Melakukan pengolahan pada data waktu		
File Target		: data/provider.py		
Last Run		: 03/07/18 09:32		
Duration		: 1,72 detik		
Test Case No.		: 002		
Test Name		: test.provider_test.TimeTestCase.test_time_process_1		
Test Case Description		: Mengubah teks menjadi waktu dengan tipe datetime, format bulan tanggal, tahun		
Assertions				
No	Assertion	Expected	Actual	Result

1	AssertEqual	Data dengan tipe datetime	Hasil pengolahan waktu dengan input teks	☑

Hasil Test Case mengambil data dengan ID dari database untuk dilakukan pengolahan teks

No.		: 202		
Scenario		: Melakukan pengolahan menggunakan Google Maps Client		
File Target		: data/provider.py		
Last Run		: 03/07/18 09:22		
Duration		: 1,59 detik		
Test Case No.		: 001		
Test Name		: test.provider_test.MapsTestCase.test_place_process		
Test Case Description		: Mengambil data dengan ID dari database untuk dilakukan pengolahan teks		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	mockup data dengan nama alamat dan posisi geometri yang sudah dicari sebelumnya	Hasil pengolahan tempat dengan input teks	☑

Hasil Test Case mengubah teks menjadi lokasi dengan format nama, alamat dan posisi geometri

No.		: 202		
Scenario		: Melakukan pengolahan menggunakan Google Maps Client		
File Target		: data/provider.py		
Last Run		: 03/07/18 09:26		
Duration		: 8,12 detik		
Test Case No.		: 002		
Test Name		: test.provider_test.MapsTestCase.test_get_data		
Test Case Description		: Mengubah teks menjadi lokasi dengan format nama, alamat dan posisi geometri		
Assertions				
No	Assertion	Expected	Actual	Result

1	AssertEqual	mockup data yang ada di database dan hasil olahannya	Hasil data yang diambil dan diolah dengan input id	☑

Hasil Test Case mengolah kumpulan token dan label dengan format B1 B2 I1 O O

No.	: 203
Scenario	: Melakukan pengolahan kumpulan token dan label menjadi entitas
File Target	: data/provider.py
Last Run	: 03/07/18 09:02
Duration	: 1,31 detik
Test Case No.	: 001
Test Name	: test.provider_test. ExtractTextTestCase.test_extract_text_0
Test Case Description	: Mengolah kumpulan token dan label dengan format B1 B2 I1 O O

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	Kumpulan data label dengan PLACE dan TIME	Hasil data yang diproses dari kumpulan teks dan kumpulan label dengan urutan B-PLACE, B-TIME, I-TIME, O, O	☑

Hasil Test Case mengolah kumpulan token dan label dengan format B1 I1 O B2 B3 O O

No.	: 203
Scenario	: Melakukan pengolahan kumpulan token dan label menjadi entitas
File Target	: data/provider.py
Last Run	: 03/07/18 09:06
Duration	: 0,86 detik
Test Case No.	: 002
Test Name	

Test Case Description		: test.provider_test. ExtractTextTestCase.test_extract_text_1 : Mengolah kumpulan token dan label dengan format B1 I1 O B2 B3 O O		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Kumpulan data label dengan NAME, PLACE dan TIME	Hasil data yang diproses dari kumpulan teks dan kumpulan label dengan urutan B-NAME, I-NAME, O, B-TIME, B-PLACE, O, O	☑

Hasil Test Case mengolah kumpulan token dan label dengan format B1 B1 O B2 I3 O O

No.		: 203		
Scenario		: Melakukan pengolahan kumpulan token dan label menjadi entitas		
File Target		: data/provider.py		
Last Run		: 03/07/18 09:09		
Duration		: 1,14 detik		
Test Case No.		: 003		
Test Name		: test.provider_test. ExtractTextTestCase.test_extract_text_2		
Test Case Description		: Mengolah kumpulan token dan label dengan format B1 B1 O B2 I3 O O		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Kumpulan data label dengan INFO, INFO dan TIME	Hasil data yang diproses dari kumpulan teks dan kumpulan label dengan urutan B-INFO, B-INFO, O, B-TIME, I-PLACE, O, O	☑

Hasil Test Case mengolah kumpulan token dan label dengan format B1 I1 O B1 I1 O O dengan teks yang sama

No.	: 203
Scenario	: Melakukan pengolahan kumpulan token dan label menjadi entitas
File Target	: data/provider.py
Last Run	: 03/07/18 09:06
Duration	: 1,14 detik
Test Case No.	: 004
Test Name	: test.provider_test. ExtractTextTestCase.test_extract_text_3
Test Case Description	: Mengolah kumpulan token dan label dengan format B1 I1 O B1 I1 O O dengan teks yang sama

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	Kumpulan data label dengan INFO	Hasil data yang diproses dari kumpulan teks dan kumpulan label dengan urutan B-INFO, B-INFO, O, B-INFO, I-INFO, O, O	☑

Hasil Test Case membuat data untuk dilabel oleh model

No.	: 301
Scenario	: Menggunakan modul Model Adapter tanpa inisiasi data
File Target	: model/adapter.py
Last Run	: 03/07/18 19:58
Duration	: 0,02 detik
Test Case No.	: 001
Test Name	: test.adapter_test.AdapterTestCase.test_for_tagging_0
Test Case Description	: Membuat data untuk dilabel oleh model

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	Data mockup yang terdiri dari daftar kumpulan token	Hasil pengubahan data yang ada pada database	☑

			menjadi kumpulan token	

Hasil Test Case membuat data untuk uji coba pada model

No.		: 301		
Scenario		: Menggunakan modul Model Adapter tanpa inisiasi data		
File Target		: model/adapter.py		
Last Run		: 03/07/18 20:00		
Duration		: 0,01 detik		
Test Case No.		: 002		
Test Name		: test.adapter_test.AdapterTestCase.test_for_testing_0		
Test Case Description		: Membuat data untuk uji coba pada model		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data mockup yang terdiri dari daftar kumpulan token beserta labelnya	Hasil pengubahan data yang ada pada database menjadi kumpulan token beserta labelnya	☑

Hasil Test Case melakukan tokenisasi pada teks

No.		: 301		
Scenario		: Menggunakan modul Model Adapter tanpa inisiasi data		
File Target		: model/adapter.py		
Last Run		: 03/07/18 20:07		
Duration		: 0,01 detik		
Test Case No.		: 003		
Test Name		: test.adapter_test. AdapterTestCase.test_for_tokenize_tag_0		
Test Case Description		: Melakukan tokenisasi pada teks		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data mockup kumpulan token	Hasil kembalian pertama pada pengubahan teks	☑

			menjadi kumpulan token	
2.	AssertEqual	Jumlah data pada mockup kumpulan label	Hasil kembalian kedua pada pengubahan teks menjadi kumpulan token	☑

Hasil Test Case melakukan tokenisasi pada teks multi baris

No.	: 301
Scenario	: Menggunakan modul Model Adapter tanpa inisiasi data
File Target	: model/adapter.py
Last Run	: 03/07/18 20:07
Duration	: 0,01 detik
Test Case No.	: 004
Test Name	: test.adapter_test. AdapterTestCase.test_for_tokenize_tag_1
Test Case Description	: Melakukan tokenisasi pada teks multi baris

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	Data mockup kumpulan token	Hasil kembalian pertama pada pengubahan teks menjadi kumpulan token	☑
2.	AssertEqual	Jumlah data pada mockup kumpulan label	Hasil kembalian kedua pada pengubahan teks menjadi kumpulan token	☑

Hasil Test Case melakukan pelabelan pada data

No.	: 301
Scenario	: Menggunakan modul Model Adapter tanpa inisiasi data
File Target	: model/adapter.py
Last Run	: 03/07/18 20:11
Duration	: 0,01 detik
Test Case No.	: 005

Test Name		: test.adapter_test.AdapterTestCase.test_for_tag_0		
Test Case Description		: Melakukan pelabelan pada data		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data kumpulan token yang diiterasi	Hasil data yang telah dilabel diiterasi kemudian diambil value pada indeks ke 0	☑

Hasil Test Case membuat data untuk dilabel oleh model

No.		: 302		
Scenario		: Menggunakan modul Model Adapter dengan inisiasi data		
File Target		: model/adapter.py		
Last Run		: 03/07/18 20:41		
Duration		: 0,01 detik		
Test Case No.		: 001		
Test Name		: test.adapter_test. AdapterWithInputTestCase.test_for_tagging_1		
Test Case Description		: Membuat data untuk dilabel oleh model		
Assertions				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data mockup yang terdiri dari daftar kumpulan token	Hasil pengubahan data yang ada pada database menjadi kumpulan token	☑

Hasil Test Case membuat data untuk uji coba pada model

No.	: 302
Scenario	: Menggunakan modul Model Adapter dengan inisiasi data
File Target	: model/adapter.py
Last Run	: 03/07/18 20:41
Duration	: 0,01 detik
Test Case No.	: 002
Test Name	: test.adapter_test. AdapterWithInputTestCase.test_for_testing_1

Test Case Description : Membuat data untuk uji coba pada model **Assertions**				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data mockup yang terdiri dari daftar kumpulan token beserta labelnya	Hasil pengubahan data yang ada pada database menjadi kumpulan token beserta labelnya	☑

Hasil Test Case melakukan pelabelan pada data

No. : 302 Scenario : Menggunakan modul Model Adapter dengan inisiasi data File Target : model/adapter.py Last Run : 03/07/18 20:43 Duration : 0,01 detik Test Case No. : 003 Test Name : test.adapter_test. AdapterWithInputTestCase.test_for_labeling_0 Test Case Description : Melakukan pelabelan pada data **Assertions**				
No	Assertion	Expected	Actual	Result
1	AssertEqual	Data kumpulan token yang diiterasi	Hasil data yang telah dilabel diiterasi kemudian diambil value pada indeks ke 0	☑
1	AssertEqual	Data kumpulan label yang diiterasi	Hasil data yang telah dilabel diiterasi kemudian diambil value pada indeks ke 1	☑

Hasil Test Case mengambil ID dari data terbaru

No. : 401 Scenario	

	: Melakukan pengambilan data menggunakan modul
File Target	Crawl
Last Run	: crawl.py
Duration	: 04/07/18 15:12
Test Case No.	: 2,41 detik
Test Name	: 001
Test Case Description	: test.crawl_test.CrawlTestCase.test_get_newest_id
	: Mengambil ID dari data terbaru

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	Tipe data integer	Tipe data hasil kembalian dari mengambil id terbaru dari data facebook	☑
2	AssertEqual	Tipe data integer	Tipe data hasil kembalian dari mengambil id terbaru dari data twitter	☑

Hasil Test Case mengambil ID dari data terbaru dengan input yang salah

No.	: 401
Scenario	: Melakukan pengambilan data menggunakan modul Crawl
File Target	: crawl.py
Last Run	: 04/07/18 15:12
Duration	: 0,92 detik
Test Case No.	: 002
Test Name	: test.crawl_test.CrawlTestCase.test_get_newest_id
Test Case Description	: Mengambil ID dari data terbaru dengan input yang salah

Assertions

No	Assertion	Expected	Actual	Result
1	AssertEqual	Tipe data integer	Tipe data hasil kembalian dari mengambil id terbaru dari data facebook	☑

2	AssertEqual	Tipe data integer	Tipe data hasil kembalian dari mengambil id terbaru dari data twitter	☑